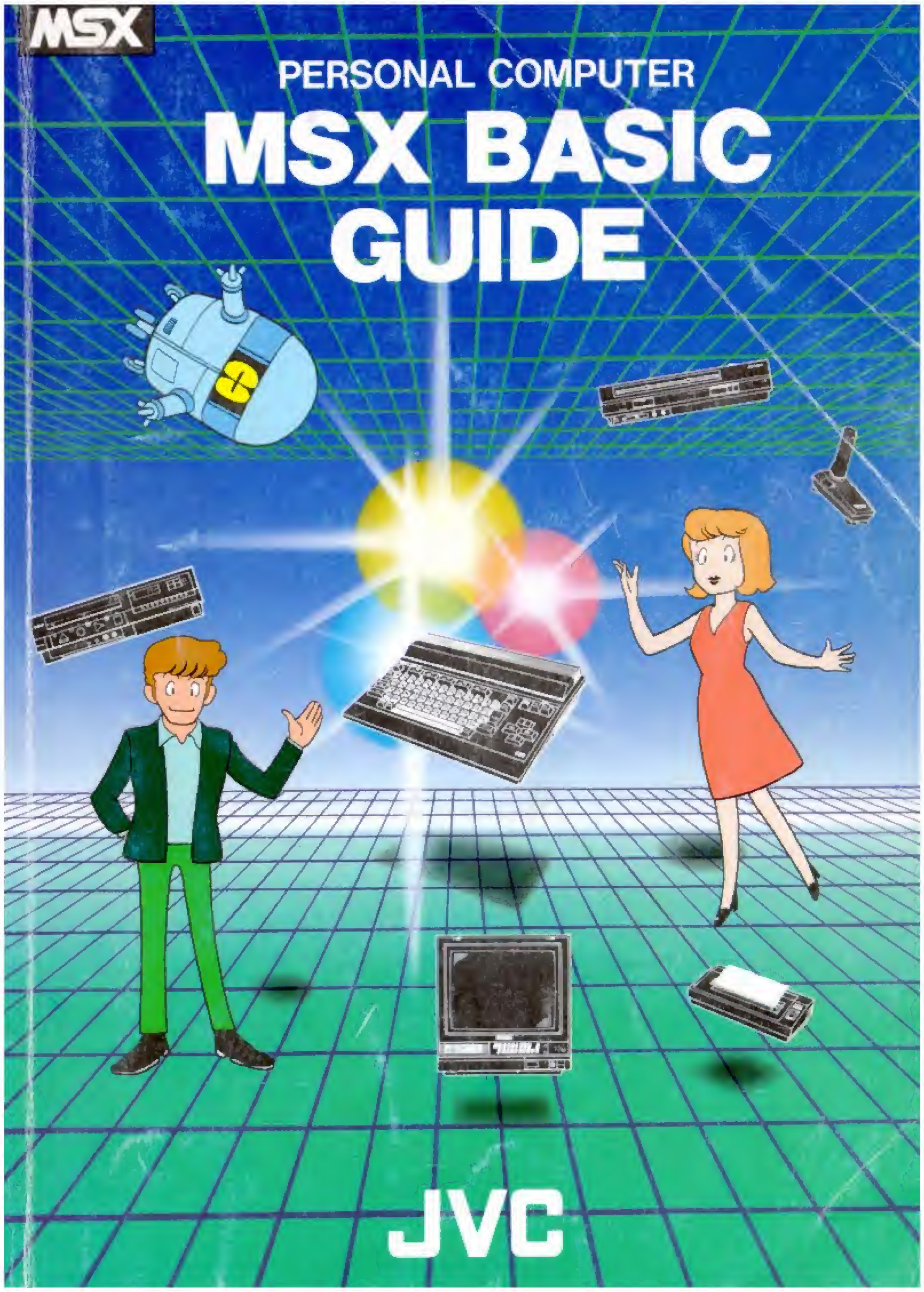
MSX

PERSONAL COMPUTER

MSX BASIC GUIDE



JVC

CONTENTS

Preface

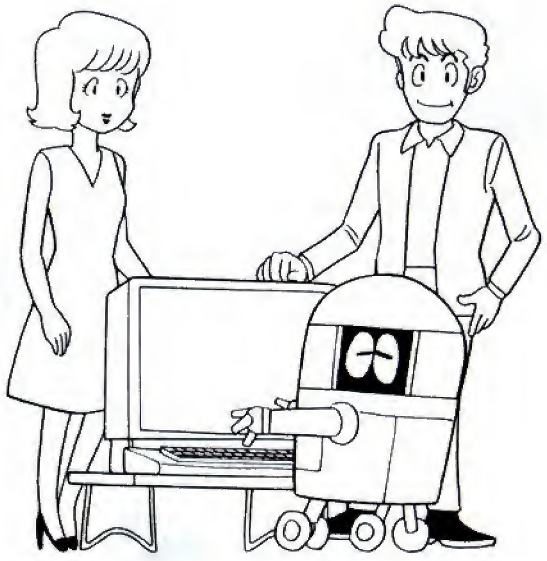
This book is written as an introduction for the beginner who will be writing BASIC programs on an MSX computer. It presents easy to understand explanations of the basic principles of MSX-BASIC programming. If you write and then run the programs in the order they are presented in, when you have finished reading this book you should be able to write simple programs by yourself, as well as modify other programs. Let's get used to BASIC by actually using the MSX computer to draw lines, produce sounds and perform other operations.

How this book is organized:

1. This book is written with the assumption that you know how to operate the computer and use the keyboard.
2. The subject matter will be much easier to understand if you enter and run each program as you read through the book.
3. The command words used in each explanation are framed in large type, so it's easy to look them up in the index.
4. Items containing a black mark • in the middle of the explanation, provide additional information to make the subject matter easier to understand.

Note:
1. If you have carefully followed the contents of this book and there are still unclear points, omissions or other mistakes in the text, please contact your sales representative for additional information.
2. Please understand that Japan Victor Corporation can not provide compensation or be responsible for any errors or problems which are the result of the items mentioned above.

Prologue
What is MSX

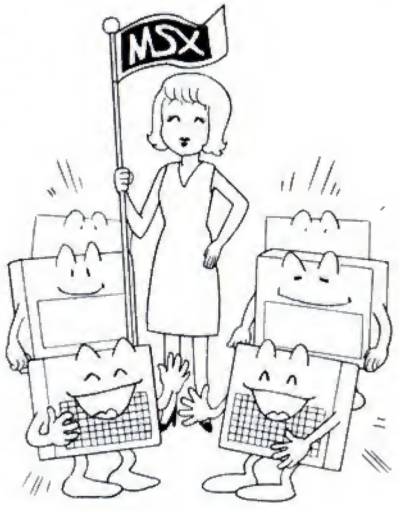


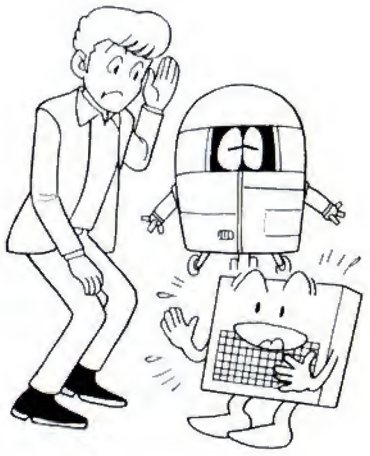
Let's speak with BASIC

It's necessary for us to use words in order to speak. In the same way, in order for people to be able to communicate with a computer, it's necessary to use a programming language such as BASIC.

Up to now, depending on either the computer or the computer manufacturer, both the BASIC and its functions have been different. As a result, when certain types of software, such as programs written in BASIC, are used on a specific computer, that software can't be used on a different kind of computer. However, on MSX computers, both the hardware and the software have been standardized. As a result, all software with the MSX logo will run on any MSX computer.

This means that you can share your software with any of your friends who also have an MSX computer.





The Key is in the Program

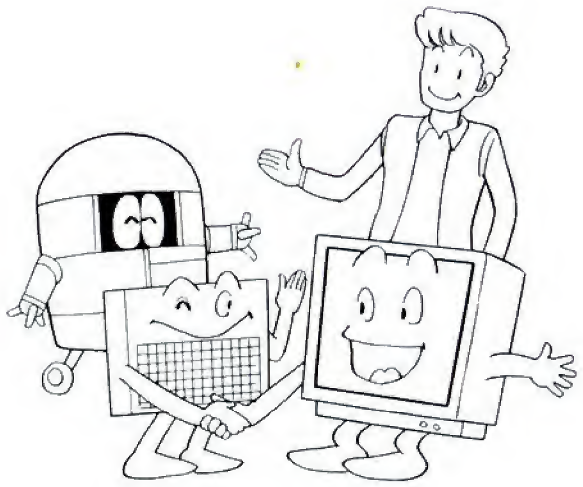
In order to make a computer run, you have to study how to write a program.

```
10  REM♥♥♥♥♥♥♥♥♥♥♥♥
20  REM♥      HC-7GB      ♥
30  REM♥♥♥♥♥♥♥♥♥♥♥♥
40  COLOR 15,4:CLS
50  LOCATE 5,2
60  PRINT "MSX Personal
    Computer"
70  LOCATE 12,5
80  PRINT "JVC"
90  LOCATE 11,7
100 PRINT "HC-7GB"
110 PLAY "CDEFE"
120 END
```

If a program is entered into the computer, it reads one line of the program at a time, and then operates according to the indicated instructions. In other words, a program is the written set of instructions which contain the appropriate commands for the computer to perform.

Well everybody, do your best to learn how to write your own programs all by yourself.

STEP 1
Give Your Computer an Order!



A program uses commands to indicate to the computer what it must do. However, there is also a way to use a computer without using a program. A computer can be run by either method. It isn't necessary to distinguish between the two as they both use the same command language.

Direct Mode

This mode is used to obtain direct results without storing the commands in the computer.

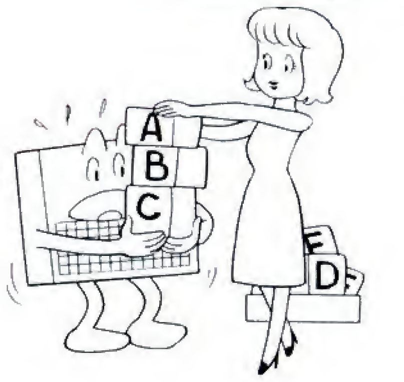
For example,

```
PRINT "ABC" [RETURN]
```

The [RETURN] key is used to inform the computer when the input is finished. Be sure to press the [RETURN] key at the end of every input line.

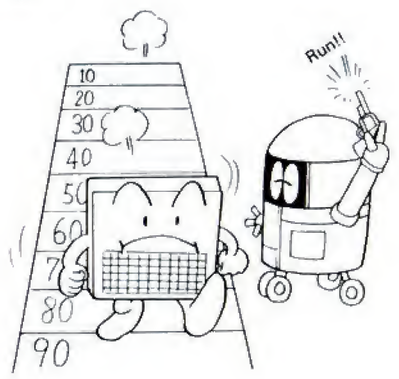
⇩

```
ABC
OK
```



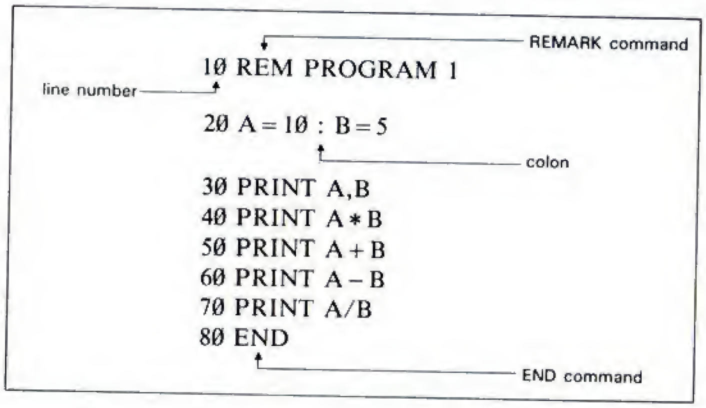
The Direct Mode
The computer can read and write characters directly.

Work is performed according to the programmed order.



Program Mode

All input lines start with a line number in the program mode. In this mode, the program is executed after the program is recorded by the computer.



```
1Ø REM PROGRAM 1
2Ø A=1Ø : B=5
3Ø PRINT A,B
4Ø PRINT A*B
5Ø PRINT A+B
6Ø PRINT A-B
7Ø PRINT A/B
8Ø END
```

Line Number

The line numbers are used to inform the computer of the correct order of the operations to be performed. Any number from Ø to 65529 may be used.

REM

The contents of the remarks statement are ignored when the program runs. They are only used to make the program easier to understand. An apostrophe (') may be used if the REM statement is omitted.

END

When the END statement is encountered, it stops the program.

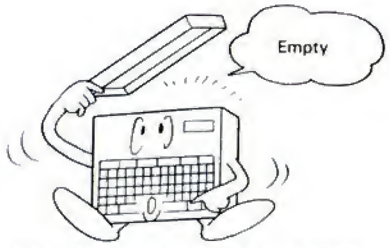
Next, let's enter this program.
Before starting to write the program the following command should be entered.

```
NEW [RETURN]
```

NEW

This command clears the entire program storage area.

If this command is not used at the beginning of a program, the program commands from the previous program may not be erased. When this happens, it's possible that program commands of the new program may be mixed with the statements already in memory.



If the old programs remain, bugs will appear.

Therefore

```
10 REM PROGRAM 1 [RETURN]
```

Don't forget to press the [RETURN] key when you finish a line.

From now on, enter all programs in this manner.

• It's convenient to memorize this command.

AUTO

This command generates the line numbers automatically on the screen.

Format

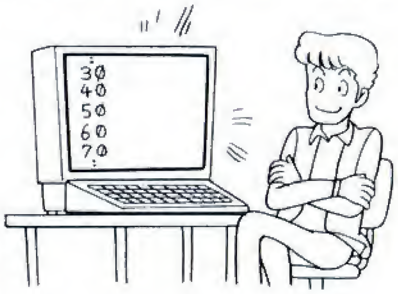
```
AUTO beginning line number, increment
```

When the first line number parameter is omitted, a value of Ø is entered. If the increment parameter is omitted a value of 1Ø is entered.

AUTO 1ØØ,1Ø	line numbers are written from line 1ØØ in increments of 1Ø
AUTO 2Ø	line numbers are written from line 2Ø in increments of 1Ø
AUTO ,1Ø	line numbers are written from line Ø in increments of 1Ø
AUTO	line numbers are written from line 1Ø in increments of 1Ø

In order to stop automatic line number generation, press the STOP key while holding the CTRL key down.

• This is a convenient command to use when lines are added to other lines, because an increment of 1Ø is often used in programs.



You don't have to take the trouble to enter a new program.

KEY

This command defines the operations of the functions keys.

Format

```
KEY function key number, character format
```

The function keys are numbered from 1 to 1Ø. A character string of up to 15 characters can be defined for any one key, but only five characters can be displayed on the screen.

```
KEY 6, "PRINT" RETURN
```

Function key F6 is defined as PRINT.

Check to see if there were any mistakes when the program was entered. If it's correct, let's try and run the program.

```
RUN RETURN
```

RUN

This command is used to execute a program in memory.

```
10 REM PROGRAM 1
20 A=10 : B=5
30 PRINT A,B
40 PRINT A*B
50 PRINT A+B
60 PRINT A-B
70 PRINT A / B
80 END
```

⇩ **RUN**

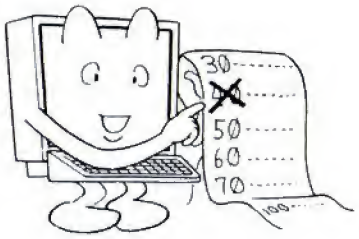
```
 10              5
 50
 15
 5
 2
OK
■
```

How did it go? Did everything run properly?

• Common Errors

```
Syntax error in line number
```

Did an error message like this appear? An error occurs when a command is used that does not appear in BASIC. Generally, typing mistakes are the most frequent reason for error messages. The last number in the error message indicates the line number where the error occurred. Find and correct the mistake in the indicated line number. Please refer to the reference book for more detailed information as there are many other possible error messages besides "Syntax Error". Let's take another look at the program by entering



An error message tells the line number of a mistake.

```
LIST  RETURN
```

LIST

The LIST command displays the program on the screen.

Format

LIST	The entire program is displayed on the screen
LIST 2Ø	Only line 2Ø is displayed on the screen
LIST 2Ø –	All the lines from line 2Ø to the end of the program are displayed on the screen
LIST –5Ø	All lines up to line 5Ø are displayed on the screen
LIST 3Ø–6Ø	Lines 3Ø to 6Ø are displayed on the screen

• How to correct a program.

(1) Rewriting a command
Suppose that you want to change the A + B in line 5Ø to A * B. Move the cursor key to the position of the "+" sign and enter the "*" symbol. Then press the RETURN key. The following characters will then appear.

```
5Ø PRINT A*B  RETURN
```

(2) Erasing an entire command line
To erase line number 7Ø enter

```
7Ø  RETURN
```

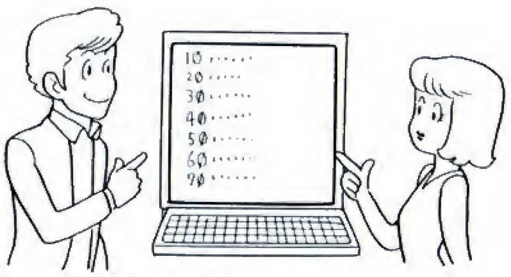
When a line number is entered and the RETURN key is pressed, the computer erases that line number.

```
1Ø REM PROGRAM #1
2Ø A=1Ø : B=5
3Ø PRINT A;B
4Ø PRINT A*B
5Ø PRINT A+B
6Ø PRINT A-B
7Ø PRINT A/B
8Ø END
```

After correcting the program, list it again.

```
4Ø PRINT A*B
5Ø PRINT A*B
6Ø PRINT A-B
8Ø END
```

The corrections which were entered are now part of the program.



There are two ways to erase commands.

• Here is another command which is convenient to memorize.

RENUM

This command is used to correct line numbers

Format

```
RENUM new line number, old line number, increment
```

If the new line number is omitted. a value of 1Ø is entered. If the old line number is omitted the first line number at the beginning of the program is entered. When the increment parameter is omitted a value of 1Ø is used.

RENUM 1,1Ø,1	Line number 1Ø is changed to 1. The line numbers after that increase by 1.
RENUM 1ØØ	The first line number is changed to 1ØØ, and the following line numbers increase by 1Ø.
RENUM 1ØØ,3Ø	Line number 3Ø is changed to 1ØØ. The line numbers after that increase by 1Ø.
RENUM ,5Ø.........	Line number 5Ø is changed to 1Ø. The lines after that increase by 1Ø.
RENUM ,5Ø,5	Line number 5Ø is changed to 1Ø. The lines after that increase by 5.

DELETE

This command erases many lines at once.

Format

```
DELETE first line number to be erased—last line number
       to be erased
```

If the first line number to be erased is omitted, a value of Ø is entered.

```
DELETE 1Ø–4Ø... Lines 1Ø to 4Ø are erased.
DELETE –1ØØ..... The lines from the beginning of the
                 program to line 1ØØ are erased.
```

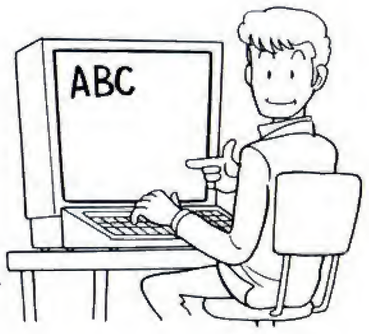
STEP 2
Let's Write a Program!

Let's study some of the basic commands necessary to write a program. Since programs are written in the following order BASIC COMMANDS, FORMAT, PROGRAM LINES AND EXPLANATIONS, when you are entering the program lines by yourself, let's also master BASIC at the same time. Be very careful about entering the comma "," semicolon ";" and other symbols. Just one mistake can make a big difference in the way the program runs.

PRINT

This command is used the most for writing letters and numbers.

PRINT "character string"	→ The characters enclosed in double quotation marks are displayed on the screen.
PRINT numerical value	→ The numerical value is displayed on the screen as shown.
PRINT	→ One line of blanks is displayed on the screen.



Character strings and answers to calculations can also be written instantly.

```
10 PRINT "ABCDEF"
20 PRINT "♥♥♥ "
30 PRINT "5*9"
40 PRINT
50 PRINT 59
60 PRINT 5*9
70 END
```

⇩ **RUN**

```
ABCDEF
♥♥♥
5*9

 59
 45
OK
■
```

When the $\boxed{\textbf{PRINT}\text{ calculation}}$ format is used, the MSX computer can be used just like an electric calculator.

```
addition        PRINT 7+8
subtraction     PRINT 15-7
multiplication  PRINT 5*8
division        PRINT 20/2
```

Let's enter the next program.

```
10 ? 7+8
20 ? 15-7
30 ? 5*8
40 ? 20/2
50 ? 200000*10000*100000
60 END
```

⇩ **RUN**

```
  15
  8
  40
  10
  2E+14
OK
■
```

Abbreviated Format

Instead of writing PRINT in each statement, the PRINT command may be omitted and a question mark "?" can be used as an abbreviation.

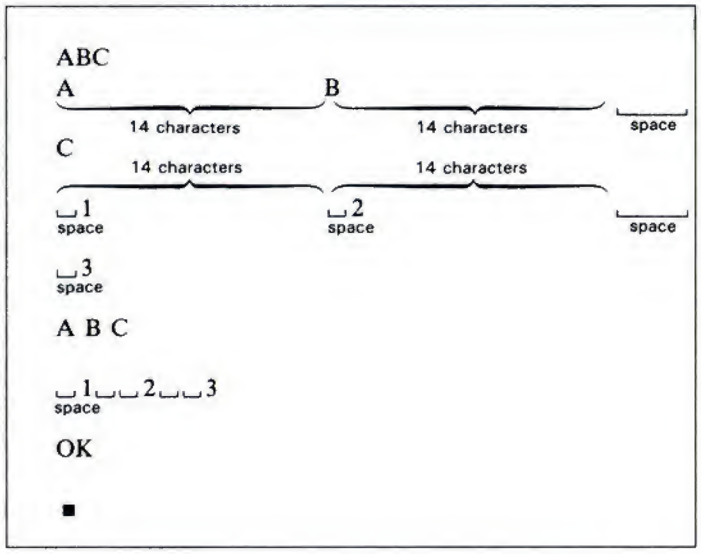
The answer for line 50 was 2E + 14. This might seem like a strange number, but this just means that the number 2 is followed by a string of 14 zeros.

✓

• Mistakes using the comma "," and the colon ";"
Enter the following program.

```
10 PRINT "ABC"
20 PRINT "A", "B", "C"
30 PRINT 1, 2, 3
40 PRINT "A" ; "B" ; "C"
50 PRINT 1 ; 2 ; 3
60 END
```

RUN



```
ABC
A             B
C
 1             2
 3
A B C
 1  2  3
OK

■
```

When the comma is used, the first 14 places of the variable number are displayed. Then the first 14 places of the second variable are displayed. When the semicolon is displayed, this indicates that the next filed to be printed will be start at the first character following the preceding field.

The first character in a numeric field is blank so that the minus sign can be printed if necessary. There is also another space at the end of a numeric field, so it will not be connected to the next field when displayed.

CLS

This command is used to clear the screen.

```
CLS
```

The clear screen command erases the contents of the screen and positions the cursor at the home position (upper left of the screen).

LOCATE

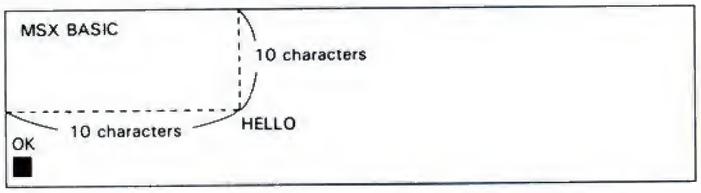
This command moves the cursor to the specified position.

```
LOCATE horizontal position, vertical position
```

The small dark square on the screen is known as the cursor. When the PRINT command is used the characters can be written in the cursor position. Because the cursor can be moved freely to any location specified by the program, characters can also be written in any desired location.

```
10 CLS
20 PRINT "MSX BASIC"
30 LOCATE 10,10 : PRINT "HELLO"
40 END
```

⇩ **RUN**



• Up to now only one command could be written on a line but in BASIC by using the colon ":" any number of commands in a single line. When a number of commands are written in a single line using the colon, this is called a multistatement. If line 3Ø is written as

```
3Ø LOCATE 1Ø,1Ø : PRINT "HELLO"
```

this is the same as

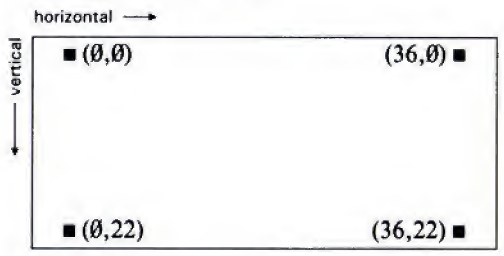
```
3Ø LOCATE 1Ø,1Ø
35 PRINT "HELLO"
```

If a program is written in multistatements the program will become shorter. The advantage of this is that the program will run faster. However, if many statements are written in the same line, the program will become harder to understand or correct, so you should avoid using this when possible.

When the power is turned on 24 lines of 37 characters can be displayed on the screen (Screen Mode Ø).

The horizontal positions are Ø – 36 (37 places).
The vertical places are Ø – 23 (24 places).

Any position within this range can be specified. This is how one page of a screen looks in the screen mode.



GOTO

Command to jump to the specified line.

Format

```
GOTO line number
```

This command causes the program to jump to the specified line number. A program is executed in the correct order specified by the program line numbers. The GOTO command is used when you want to change the order of execution.

```
10 A=1
20 PRINT A
30 A=A+1
40 GOTO 20
50 END
```

⇩ **RUN**

```
1
2
3
4
5
6
7
8
:
:
:
```

Do not use this command until you learn how to interrupt execution of a program.

```
10 A=1
20 PRINT A
30 A=A+1
```

The character "A" used here is called a variable number.

Variable: A variable is used as a container or storage area for a numerical value which can change. Please remember the name which is attached to the container or storage area.

Format ①

```
Variable name = expression
```

In this case the equals sign doesn't mean "equals" or "is the same as". Instead, this means that the value on the right is entered into the value on the left.

```
10 A=1
```

In this case, 1 is entered to the container or storage area called "A"

Format ②

```
PRINT variable name
```

```
20 PRINT A
```

In this case the value entered in the storage area called "A" is displayed on the screen.

```
30 A=A+1
```

This statement means that 1 is added to the value of A. This is entered into the storage area A and becomes the new value of A.

The first character of a variable must be an alphabetic character. After that either letters or numbers up to 255 can be used. Variable names of more than three characters can't be distinguished by the computer.

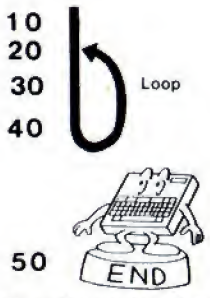
```
A123─┐
A124 │ These are all interpreted as the same variable
A125─┘
```

Only the first two characters can be distinguished by the computer.

```
40 GOTO 20
```

This means jump to line 2Ø. When this program is run, numbers will be displayed one after another on the screen without stopping.

The computer will execute the lines up to line 4Ø one by one. The program never reaches the line 5Ø END statement because when it gets to line 4Ø it jumps back to line 2Ø.



To stop execution, read the explanation of STOP on the next page.

STOP

This command is used to interrupt the program execution.

To stop the program, press the STOP key while holding the CTRL key down.

The message

```
Break in line number
```

will be displayed on the screen to indicate the point in the program where the interrupt occured.

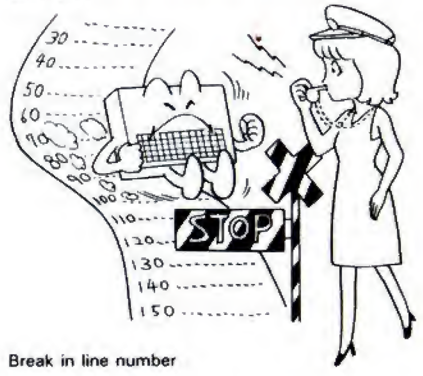
```
  .
  .
  .
  .
  .
  49
  5Ø
  51
Break in 2Ø
OK
■
```

At this time program execution can be continued by using the CONT command.

```
  .
  .
  .
  .
  49
  5Ø
  51
Break in 2Ø
OK
CONT   RETURN

  52
  53
  54
  55
  56
  .
  .
  .
  .
  .
  .
```

The interrupted line number will be stored and used later.



Break in line number

FOR ~ NEXT

Command to repeat the same process

Format

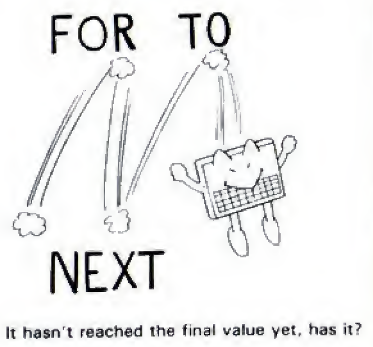
```
FOR      variable = beginning value TO ending value
 ⌇
NEXT     variable
```

This command is used to repeat the same process within a program a fixed number of times, and then stop automatically.

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
40 END
```

⇩ **RUN**

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
OK
■
```



The same variable name must be used in the FOR and the NEXT statement.
When the value of a variable is changed in constant steps, the size of each step can be specified with the STEP command.

```
STEP variable increment
```

is written like this

```
10 FOR I=1 TO 10 STEP 2
20 PRINT I
30 NEXT I
40 END
```

⇩ **RUN**

```
 1
 3
 5
 7
 9
OK
■
```

This program will display the numbers from 1 to 10 on the screen. When the STEP 2 is entered, the numbers displayed on the screen will increase by 2 each time.

If there are many steps in a subroutine, the number of written repetitions can be reduced.

INPUT

Command to assign data input from the keyboard to a variable.

Format

```
INPUT variable
```

Up to now, the following method has been used to inform the computer of a variable value.

```
10 A=10
20 B=20
      .
      .
      .
      .
```

At the beginning of the program where the variable values are defined. This is used when you want to change the values of the variables. The program has to be corrected.

The INPUT command is used to input variable values from the keyboard.

```
10 INPUT A
20 PRINT A*3
30 END
```

⇩ **RUN**

```
? ■
```

The question mark will appear. This means that the computer is asking you what the value for the variable A in the line 1Ø should be. For example, if you enter

```
? 5   RETURN
```

The computer will perform the calculation on line 2Ø as 5 * 3. The answer will be 15.

Using the INPUT command

```
? ■
```

is output, but you have no way of knowing whether character or numeric information is requested, or how you should answer. If characters enclosed in quotation marks ("") follow the INPUT command, those characters will be displayed before the question mark.

Format

```
INPUT "character string";variable
```

Well? let's change line 10 in the last program and then try and run the program

```
1Ø INPUT "HOW MUCH IS A";A
2Ø PRINT A*3
3Ø END
```

⇩ **RUN**

```
HOW MUCH IS A?  ■
```

IF ~ THEN

According to a condition

Format ①

```
IF condition THEN statement
```

If a specified condition is met, the operations following the THEN statement are performed. If the condition is not met, the next line in the program is executed.

Format ②

```
IF condition THEN line number
```

If a condition is met, the program will jump to the specified line number. If the condition is not met the next line in the program is executed.

The IF ~ THEN command is used when conditional judgements have to be performed.

```
10 INPUT A
20 IF A>50 THEN PRINT "GREATER THAN 50"
   :GOTO 10
30 PRINT "LESS THAN 50"
40 GOTO 10
50 END
```

When this program is run, and the question mark appears, enter any number and press the RETURN key.

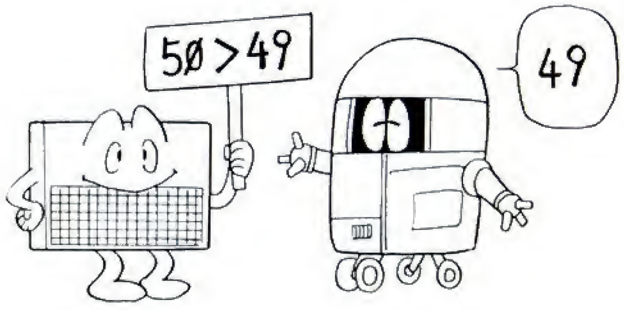
When the number which is input is greater than 50,

```
GREATER THAN 50
```

appears. When the number which is input is less than 5Ø, the following message is displayed.

```
LESS THAN 5Ø
```

Conditional format		
Symbol	Expression	Meaning
=	A=B	A and B are equal
>	A>B	A is greater than B
<	A<B	A is smaller than B
either < > or > <	A< >B	A and B are not equal
either > = or = <	A> =B	A is greater than B
either < = or = <	A< =B	A is less than B



IF ~ THEN ~ ELSE

If a condition is not satisfied, an other operation is performed.

Format ①

```
IF condition THEN command (1) ELSE command (2)
```

If a condition is met the first command is performed. If the condition is not met the second command is performed.

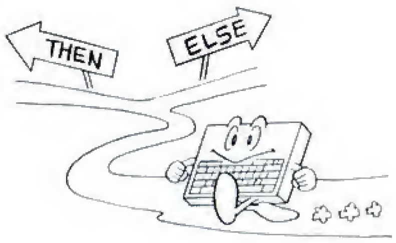
Format ②

```
IF condition THEN line number(1) ELSE line number(2)
```

If the condition is met the first line is jumped to. If the condition is not met, the program control jumps to the second line.

In this command either line numbers or commands may be used in the same line without any problem. For example `IF condition THEN command ELSE line number`. This command is now used in th preceding program.

```
10 INPUT A
20 IF A>50 THEN PRINT "GREATER THAN 50"
   ELSE PRINT "LESS THAN 50"
30 GOTO 10
40 END
```



GOSUB ~ RETURN

This command is used to jump to a subroutine.

Format ①

```
GOSUB line number
  ≀
RETURN
```

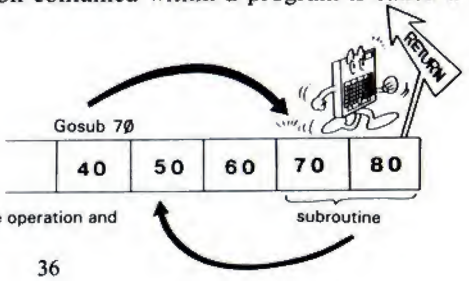
After jumping to the specified line number and performing the subroutine, by using the RETURN statement in the subroutine, the program returns to the line following the GOSUB statement.

Format ②

```
GOSUB line number
  ≀
RETURN line number
```

This is similar to the previous example. The program branches to the line number and performs the subroutine, but the program returns to the line number specified in the RETURN statement.

- A subroutine is a part of a program which repeats the same operations any number of times when the program is executed. This is extremely convenient as it can be called whenever it is needed.
  This small operation contained within a program is called a subroutine.



A subroutine does a separate operation and then returns.

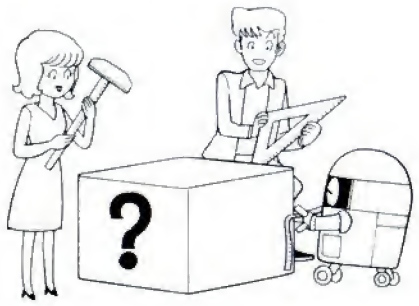
The next program calculates the surface area of a block.

```
10 INPUT "HEIGHT";A
20 INPUT "WIDTH";B
30 INPUT "DEPTH";C
40 GOSUB 70
50 PRINT "SURFACE AREA";S
60 END
70 S=2*(A*B+A*C+B*C)
80 RETURN
```

When the program is run, and the computer will ask for the values for HEIGHT? WIDTH? and DEPTH?, enter 4, 5 and 6. After line 70 a subroutine is performed. The program returns to line 50 after the RETURN in line 80, and then displays the calculation.

```
SURFACE AREA 148
```

No matter how large the rectangle is, the calculation for the surface area is the same. Let one subroutine do this calculation.



READ ~ DATA

This command is used to enter data.

Format

```
READ variable 1, variable 2, variable 3...
```

Data is entered as represented by variable (1), variable (2) and variable (3).

```
DATA constant(1), constant(2), constant(3)
```

More than one constant can be contained in a DATA statement.

When the variable value is entered, a number of fixed constants can be entered.
When the variable value is entered, the INPUT statement is used to represent values such as A = 1Ø, B = 2Ø etc.
However, there is also another way to enter data besides this method. This is done by using the READ ~ DATA statements.

```
1Ø READ X,Y,Z
2Ø PRINT X;Y;Z
3Ø DATA 1Ø,2Ø,3Ø
4Ø END
```

⇩ **RUN**

```
 1Ø  2Ø  3Ø
OK
■
```

In line 2Ø the READ X, Y, Z statement reads and enters (loads) the values of X, Y and Z. The values entered for X, Y and Z correspond to the values in either lines 1Ø, 2Ø, or 3Ø.

The READ statement variables can have either a numeric value or a character variable.

For example in the next program, the average test scores for Tom, Dick and Harry can be calculated.

```
1Ø FOR I = 1 TO 3
2Ø READ A$, B, C, D
3Ø X=INT((B+C+D)/3)
4Ø PRINT A$; "AVERAGE SCORE ";X
5Ø NEXT
6Ø END
7Ø DATA Tom, 74, 69, 78
8Ø DATA Dick, 66, 55, 61
9Ø DATA Harry, 87, 76, 73
```

⇩ **RUN**

```
Tom AVERAGE SCORE 73
Dick AVERAGE SCORE 6Ø
Harry AVERAGE SCORE 78
OK
■
```

```
10 FOR I=1 TO 3
50 NEXT I
```

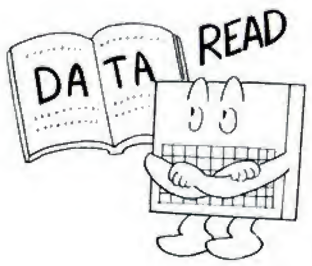
The FOR-NEXT commands in lines 10 and 50 will be repeated three times. Three repetitions are necessary to perform the calculations for each of the three students.
Line 20 reads each student's name and grades for English, mathematics and science. This information is contained in the DATA statements in lines 70, 80 and 90.

```
30 X=INT((B+C+D)/3)
```

This statement adds the three grades and then divides them by 3 to get the average.
The INT command is used to obtain the positive integral value (i.e. without any fractions) after the sum of B + C + D is divided by 3. We'll study the INT command in more detail later.

```
40 PRINT A$; "AVERAGE ";X
```

The name which was loaded in line 20 and the average calculated in line 30 are displayed.



The calculations are performed in the same manner, but the READ-DATA statement, makes it easy to enter a number of variables.

RESTORE

This command is used to read and enter data after it has already been read in one time.

Data which was loaded by the READ ~ DATA commands normally can't be read again after it has been loaded once. However, it is often necessary to use the same data a number of times. The RESTORE command enables the data to be loaded again after it has already been loaded once.

Format ①

```
RESTORE
```

```
10 READ A, B, C
20 RESTORE
30 READ D, E, F
40 PRINT A;B;C
50 PRINT D;E;F
60 END
70 DATA 10, 20, 30
```

⇩ **RUN**

```
 10  20  30
 10  20  30
OK
■
```

Format ②

```
RESTORE line number
```

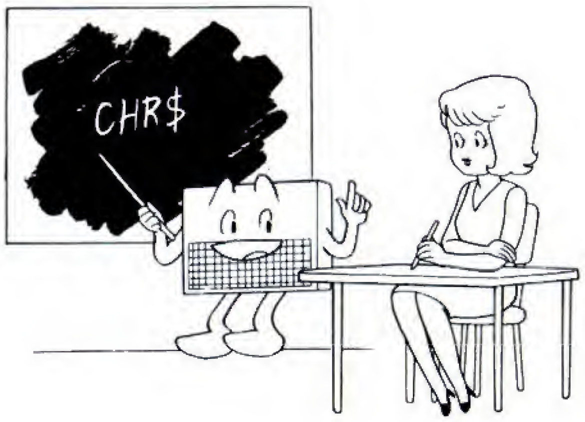
When this command is used the data in the specified line number will be loaded next.

```
10 READ A, B, C
20 RESTORE 90
30 READ D,E,F
40 PRINT A;B;C
50 PRINT D;E;F
60 END
70 DATA 10, 20, 30
80 DATA 40, 50, 60
90 DATA 70, 80, 90
```

⇩ **RUN**

```
 10  20  30
 70  80  90
OK
■
```

STEP 3
Are Functions Difficult?



The way functions work is rather simple. An answer is given for a value based on a preset processing operation. MSX computers have many functions, each of which performs a preset process.
Lets study how to use some of the more useful functions.

Preset processing of variables is efficient.

CHR$ function

This returns the character which corresponds to the numeric or variable value.

Format

```
CHR$ (numeric value)
```

```
CHR$ (variable value)
```

The variable value must be between Ø to 255.

```
PRINT CHR$(65)  RETURN
```

⇩

```
A
OK
■
```

The character "A" is displayed on the screen. This is because the 65th character in the character code chart is "A".
The computer can also display other characters besides the ones on the keyboard. The chart of the assigned numbers for characters and graphics is contained within the computer. This number which is assigned to a character is called the character code or ASCII code.

Character Code Chart

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0		┼		0	@	P		p	Ç	É	á	Ã	▄	◀	α	≡
1	☺	┴	!	1	A	Q	a	q	ü	æ	í	ã	▚	⧗	β	±
2	☻	┬	"	2	B	R	b	r	é	Æ	ó	Ĩ	█	⧓	Γ	≥
3	♥	┤	#	3	C	S	c	s	â	ô	ú	ĩ		▪	Π	≤
4	♦	├	$	4	D	T	d	t	ä	ö	ñ	Õ	▬	▐	Σ	⌠
5	♣	┼	%	5	E	U	e	u	à	ò	Ñ	õ	■		σ	⌡
6	♠	│	&	6	F	V	f	v	á	û	ª	Ũ	▌	▖	μ	÷
7	·	─	'	7	G	W	g	w	ç	ù	º	ũ	▞	▒	γ	≈
8	◘	┌	(	8	H	X	h	x	ê	ÿ	¿	Ĳ	█	∆	Φ	○
9	○	┐	)	9	I	Y	i	y	ë	Ö	⌐	ĳ	▕	‡	θ	●
A	◙	└	*	:	J	Z	j	z	è	Ü	¬	¾	█	ω	Ω	-
B	♂	┘	+	;	K	[	k	¦	ï	¢	½	~	▨	█	δ	√
C	♀	╳	,	<	L	\	l	¦	î	£	¼	◇	▧	▄	∞	ⁿ
D	♪	╱	-	=	M	]	m	¦	ì	¥	¡	‰	▼	▌	φ	²
E	♫	╲	.	>	N	^	n	~	Ä	Pt	«	¶	▲	▐	ε	▌
F	☼	┼	/	?	O	_	o	△	Å	ƒ	»	§	▶	▀	η	

The character code in the chart above is determined as follows.
(horizontal column number) × 16 + (vertical column number) = (assigned character code)

INT

This returns the integer and removes any fractions. This command is used for positive numbers.

Format

```
INT (numeric value)
```

```
PRINT INT (3.14)   RETURN
```

The fraction of 3.14 is removed and the positive number 3 is displayed.

RND

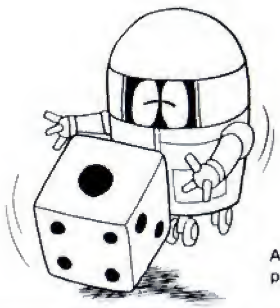
This command returns a random number between 1 and Ø.

Format

```
RND (numeric value =  -1, Ø or 1)
```

```
1Ø FOR I=1 TO 5
2Ø X=INT(RND(1)*1Ø)
3Ø Y=INT(RND(1)*1Ø)
4Ø PRINT "X=";X;"Y=";Y
5Ø NEXT I
6Ø END
```

When this program is run X is equal to a value ranging from Ø to 9. Y is equal to a value from 1 to 1Ø. This operation will write out the results five times.



Any number of random probability can be output just like throwing dice.

INKEY$

This command is used to represent characters input from the keyboard.

Format

```
INKEY$
```

If a key on the keyboard is pressed, the value of that character is returned. If no key on the keyboard is pressed, a space (null string) is returned for that value.

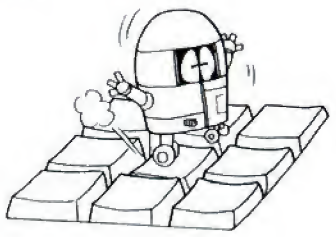
```
10 A$=INKEY$
20 IF A$=" " THEN GOTO 10
30 PRINT A$
40 END
```

When this program is run, press any key on the keyboard. The character which was entered will be displayed and the program will end.

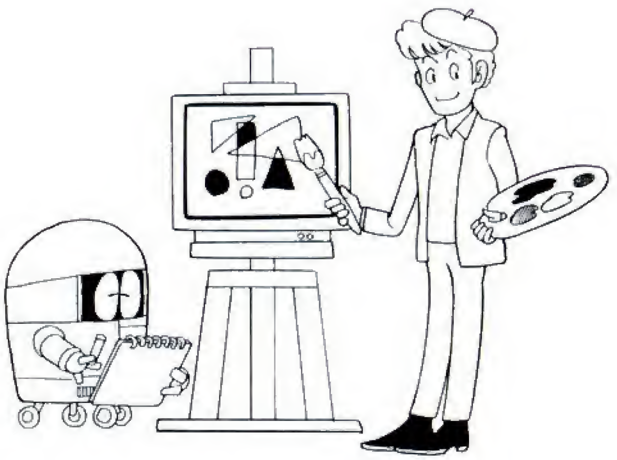
```
 10 X=10:Y=10:CLS
 20 A$=INKEY$
 30 IF A$="S" THEN X=X-1:GOTO 70
 40 IF A$="D" THEN X=X+1:GOTO 70
 50 IF A$="E" THEN Y=Y-1:GOTO 90
 60 IF A$="X" THEN Y=Y+1:GOTO 90
 70 IF X<0 THEN X=0
 80 IF X>28 THEN X=28
 90 IF Y<0 THEN Y=0
100 IF Y>20 THEN Y=20
110 LOCATE X,Y:PRINT " ♥ "
120 GOTO 20
130 END
```

This program displays the " ♥ " on the screen. If "S" is entered it will move to the left side. If "D" is entered it will move to the right side. If "E" is entered it will move up. If "X" is entered it will move down.

Program flow can be controlled from the keyboard.



STEP 4
Let's Paint a Picture with the MSX !



Let's draw pictures with the MSX!
Lets's study some commands and draw some pictures now. We'll combine some of the commands that have appeared before to create some graphics.

SCREEN

This command selects the screen.

Format

```
SCREEN mode
```

The screen mode is specified by values from Ø to 3.

MSX computer has four kinds of screen modes in all.

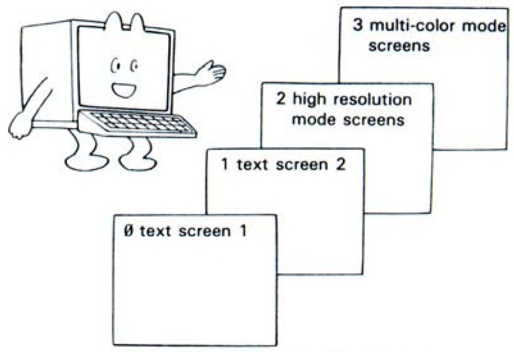
Mode Ø:	Text Screen 1 37 characters × 24 lines (Default setting)
Mode 1:	Text Screen 2 29 characters × 24 lines (Default setting)
Mode 2:	High Resolution Graphic Mode 256 dots × 192 dots
Mode 3:	Low Resolution Graphic Mode (Multi-color Mode) 256 dots × 192 dots (64 × 48 blocks)

Up to now we have only used Screen Mode Ø.
The screen is automatically set to Mode Ø when the power is turned on.
The Text Mode and Graph Mode are used as follows.

To display characters	Text Mode
To display drawings	Graphic Mode

Use Text Screen 1 or Text Screen 2 to display any character on the screen

Mode 2 or Mode 3 must be used to display any graphics.



You have just been using Text screen 1 of mode Ø. There are three other kinds of screen modes.

COLOR

This command is used to select colors for the screen.

In MSX BASIC there are 16 colors which may be used. But they are decided by the color command.

Format

```
COLOR character color code, background color code,
      border color code
```

This command determines the color of characters and lines,the background and border of the screen.

When the computer is turned on it is preset to

```
COLOR 15, 4, 4
```

White characters (color code 15)
Dark blue background (color code 4)
Blue border (color code 4) — Valid only in Mode 1.

The border area is not displayed because the screen in Mode Ø. Change the color of the screen by refering to the Color Code Chart and making the appropriate modifications.

• Color Code Chart

0: clear	4: dark blue	8: red	12: dark yellow
1: black	5: light blue	9: light red	13: purple
2: green	6: dark red	10: yellow	14: gray
3: light green	7: blue	11: light yellow	15: white

• Abbreviated Format
To change just the character color

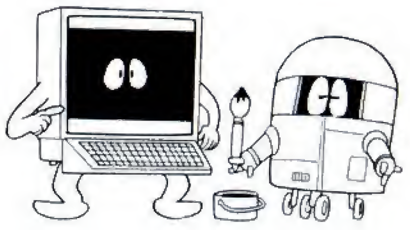
```
COLOR 1
```

To change just the background color

```
COLOR ,2
```

To change the outline color

```
COLOR,,3
```



I made a mistake. The characters and background were the same color.

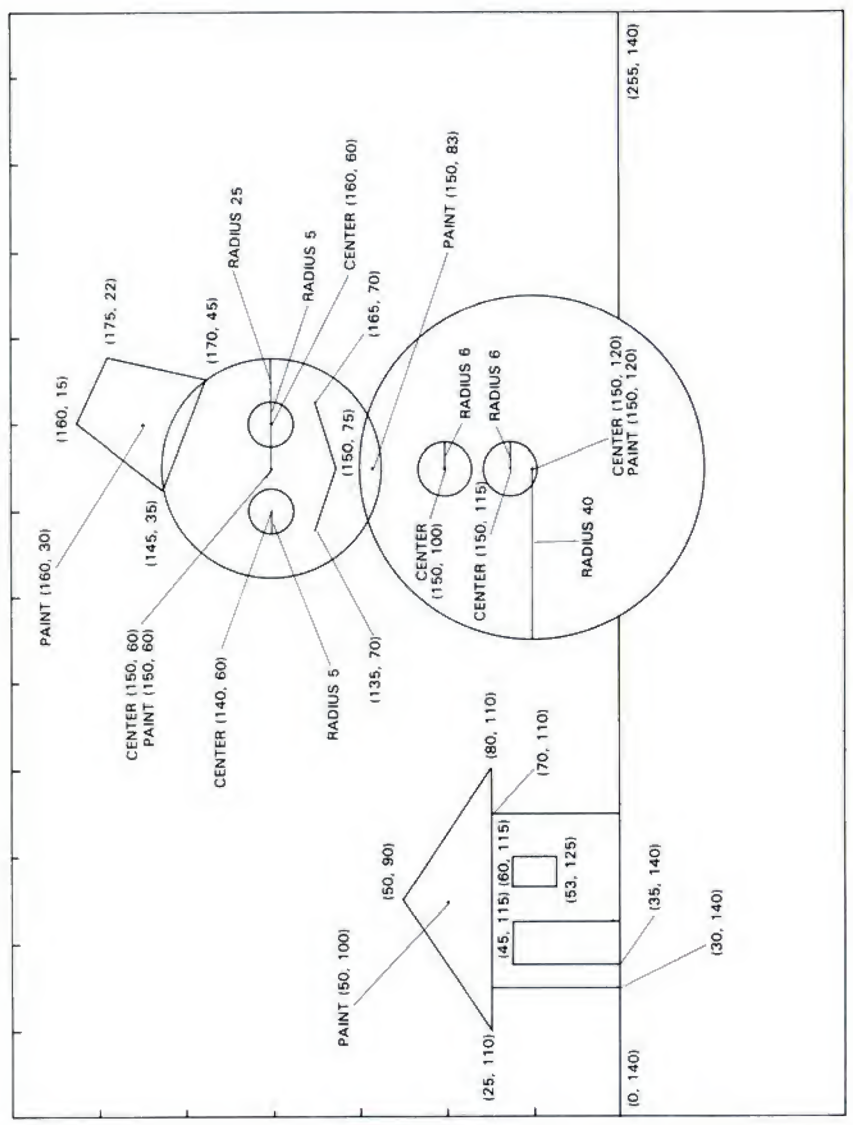
Well, let's try and paint a picture with the MSX computer and the next program.

This program used circles and straight lines to make a snowman and a house. Now let's make the snow fall in the sky.

```
 10 REM SNOWMAN                          } Program title set mode
 20 SCREEN 2                             } set mode
 30 LINE (0,140)-(255,140), 5            } background sky
 40 PAINT (70,40), 5                     }
 50 LINE (25,110)-(50,90), 6             }
 60 LINE (50,90)-(80,110), 6             } roof of the house
 70 LINE (80,110)-(25,110), 6            }
 80 PAINT (50,100), 6                    }
 90 LINE (30,140)-(70,110), 10, BF       } house
100 LINE (35,140)-(45,115), 1, BF        } door
110 LINE (53,125)-(60,115), 3, BF        } window
120 CIRCLE (150,120), 40,15              }
130 CIRCLE (150,60), 25,15               }
140 PAINT (150,120), 15                  } snowman
150 PAINT (150,60), 15                   }
160 PAINT (150,83), 15                   }
170 CIRCLE (140,60), 5,1                 }
180 CIRCLE (160,60), 5,1                 } eyeballs
190 PAINT (140,60), 1                    }
200 PAINT (160,60), 1                    }
210 LINE (135,70)-(150,75), 1            } mouth
220 LINE(150,75)-(165,70), 1             }
230 CIRCLE (150,100), 6,9                }
240 CIRCLE (150,115), 6,9                } buttons
250 PAINT (150,100), 9                   }
260 PAINT (150,115), 9                   }
270 LINE (145,35)-(160,15), 13           }
280 LINE (160,15)-(175,22), 13           }
290 LINE (175,22)-(170,45), 13           } bucket on head
300 LINE (170,45)-(145,35), 13           }
310 PAINT (160,30), 13                   }
```

```
320 FOR I=1 TO 500              ⎫
330 X=INT(RND(1)*256)           ⎬ place dots in sky
340 Y=INT(RND(1)*91)            ⎪ at random
350 PSET (X, Y), 15             ⎪
360 NEXT I                      ⎭
370 GOTO 370                    } continue displaying
                                  the picture
380 END
```

To paint a line with the MSX computer, the position of the dots on the screen must be displayed one by one. In order to paint the picture of a snowman, the positions of the necessary points are drawn in a diagram.



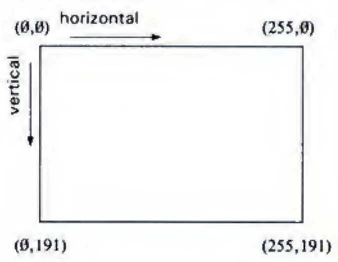
Did you draw it correctly? If you make a mistake in the numbers, the snowman might not have a head, or the inner colors might spread all over the screen.

SCREEN

Selection for line drawing

```
20 SCREEN 2
```

This selects a high resolution mode of 256 (horizontal) × 192 (vertical) dots. As a result there are 256 individual horizontal points and 192 vertical points which can be painted. These are known as graphic coordinates.



If you're thinking that it's 256 and not 255, make sure that you're counting the zero.

LINE

Command for drawing a straight line

```
30 LINE(0,140)-(255,140), 5
```

This command draws a line from point A (0,140) to point B(255,100) in color number 5 (light blue).

Format

```
LINE (starting point) – (ending point), color code
```

When the color code is omitted, the color specified for the character code is used.

When a line is drawn by the LINE statement, it can also be abbreviated as follows.

```
50 LINE (25,110)–(50,90), 6
60 LINE (50, 90)–(80, 110), 6
70 LINE (80, 110)–(25, 110), 6
```

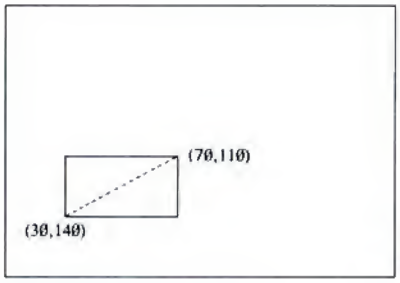
⇩

```
50 LINE (25, 110)–(50,90), 6
60 LINE –(80,110), 6
70 LINE –(25,110), 6
```

Command for painting a rectangle

```
90 LINE (30,140)–(70,110), 10, BF
```

In this command a diagonal line is drawn from point A (30,140) to point B (70,110) and connected. This forms the diagonal of a rectangle which will be painted yellow (color mode 10). The inside of this rectangle will be painted yellow.



You can draw a rectangle by just specifying 2 points.

Format

```
LINE (starting point) – (ending point), color code, B
```

Either B or BF can be used to specify the painting of the inside of the rectangle.

PAINT

Command to paint colors

```
40 PAINT (70,40), 5
```

This command will paint the area from the point (70,40) in light blue (color code 5).
Be careful when you use this command, because if the area to be painted isn't surrounded by lines, the entire screen will be filled in with the specified color.

The color will flow out if the outline isn't completely connected.

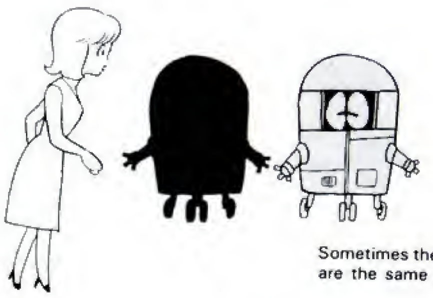
Format

```
PAINT (starting point), color, color of high resolution
```

The high color resolution can't be used on SCREEN 2.
The color code designations for painting SCREEN 2 graphics are not the same. The colors for SCREEN 3 graphics must be specified separately.

```
10 SCREEN 2
20 LINE(100,10)-(50,100), 1
30 LINE-(200,150), 1
40 LINE-(100,10), 1
50 PAINT(100,80), 1
60 GOTO 60
70 END
```

This program paints a black triangle when it is run.
After the triangle is drawn, it then paints the inside portion black.



Sometimes the color of the outline and the inside are the same and sometimes they're different.

```
10 SCREEN 3
20 LINE(100,10) - (50,100), 1
30 LINE - (200,150), 1
40 LINE - (100,10), 1
50 PAINT(100,80),2,1
60 GOTO 60
70 END
```

When this program is run it paints a black triangle.
After the triangle has been drawn, it paints the inside portion green.

CIRCLE

Command for drawing an ellipse

```
120 CIRCLE(150,120), 40,15
```

This command draws an ellipse with a center located at position (150,120). The radius is 40 dots (pixels), and the ellipse will be painted white (color code 15).

Format

```
CIRCLE (center point), radius length (in dots), color code
```

```
10 SCREEN 2
20 FOR A = 10 TO 60 STEP 10
30 CIRCLE(128,96), A, 1
40 NEXT A
50 GOTO 50
60 END
```

When this program is run, it draws six black ellipses with different radii each having the same center.
In line 5Ø the statement GOTO 5Ø causes the ellipses which were drawn to be displayed continuously. To stop the display press the STOP and CTRL keys.

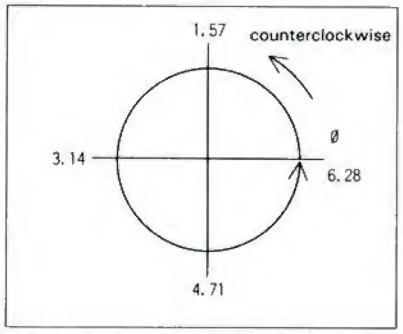
A circle is defined by the location of the center and the length of the radius. If you don't specify the color, it will be the same as the text characters.

The correct format for the circle command is shown below.

CIRCLE (center point), radius length (in dots), color code, starting point degree, ending point degree, ratio (of radius length to width).

The parameters from the color code to the ratio may be omitted.
- When the color code parameter is omitted, the color specified in the COLOR command is used.
- When the start point degree parameter is omitted a value of Ø is used.
- When the end point degree parameter is omitted a value of 6.28 is used.
- A value of 1 is used when the ratio parameter is omitted.
- Set the ratio to 1.3 to draw a circle.

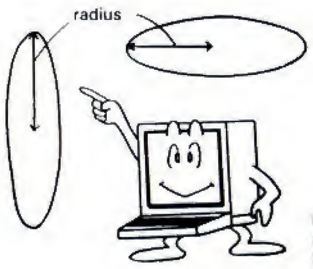
The start point angle is defined as the first position where the circle will be drawn from.
The end point angle is defined as the last position where the circle will be drawn to.



When you only want to draw 90° of a circle the end point is calculated as follows.
3.14 × (90°/180°) = 1.57

The rate is the ratio of the horizontal axis to the vertical axis of a circular geometric figure.
• A value of 1 is a circle.
• A value of less than 1 causes a horizontal elipse to be drawn.
• A value greater than 1 causes a vertical elipse to be drawn.

When the ratio is less than 1, a horizontal elipse is displayed. When the ratio is greater than 1, a vertical elipse is displayed.



Remember that when the ratio is less than 1, the radius specification represents the maximum length of the radius.

```
10 SCREEN 2
20 CIRCLE(60,50), 40,1
30 CIRCLE(80,150), 50,1,-1.57,-3.14,1
40 CIRCLE (160,100), 60,1,,,2
50 GOTO 50
60 END
```

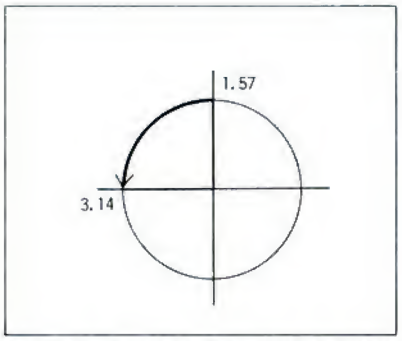
Line 20 draws the circle.

```
CIRCLE (60,50), 40, 1 , 0 , 6.28, 1
```

The parameters within the broken line can be omitted.

In line 30 the arc shown below is drawn.

```
CIRCLE (80,150), 50,1,-1.57,-3.14,1
```



The minus sign connects the center of the circle with the start and end points.

In line 40 a circle is drawn.

```
CIRCLE(160,100), 60, 1, 0, 6.28 ,2
```

The parameters inside the broken lines may be omitted.

PSET

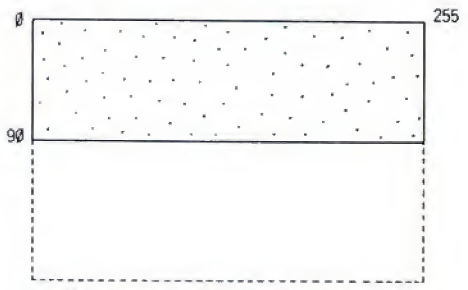
Command to draw a dot

```
350 PSET (X,Y), 15
```

This command draws a white dot at the graphic coordinates (X,Y). When no value is specified, X is preset to 320 and Y to 360.

```
320 FOR I=1 TO 500
330 X=INT(RND(1)*256)
340 Y=INT(RND(1)*91)
350 PSET(X,Y), 15
360 NEXT I
```

These commands draw 500 white dots at the graphic coordinates (256,90). The first value indicates the horizontal position, and the second value indicates the horizontal position.



The falling snow can be colored black or red by changing the color code.

Format

```
PSET (horizontal value , vertical value), color code
```

PSET

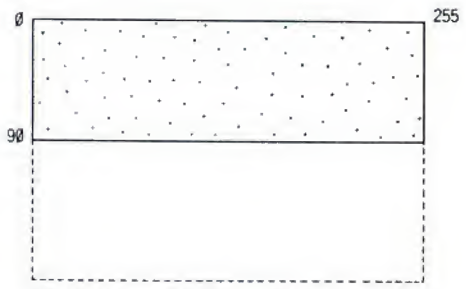
Command to draw a dot

```
350 PSET (X,Y), 15
```

This command draws a white dot at the graphic coordinates (X,Y). When no value is specified, X is preset to 320 and Y to 360.

```
320 FOR I=1 TO 500
330 X=INT(RND(1)*256)
340 Y=INT(RND(1)*91)
350 PSET(X,Y), 15
360 NEXT I
```

These commands draw 500 white dots at the graphic coordinates (256,90). The first value indicates the horizontal position, and the second value indicates the horizontal position.



The falling snow can be colored black or red by changing the color code.

Format

```
PSET (horizontal value , vertical value), color code
```

PRESET

Command to erase a point at a specified position.

Format

```
PRESET (horizontal value , vertical value), color code.
```

If a color code other than the specified background is specified, it will be the same as the PSET color code.

This is O.K.
All the commands which were used in the program "Snowman" have been explained.
Enter the characters RUN.

DRAW

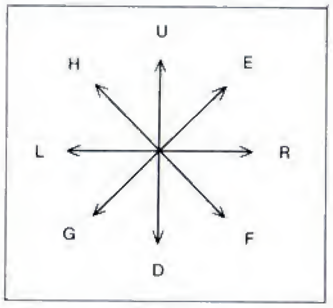
Command to set a point or extend a line.

Format

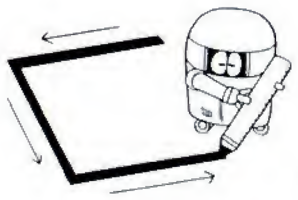
```
DRAW "character string"
```

A point moves according to the specified character string and a figure is drawn.

The DRAW command position is directed as specified by the characters U, D, L or R. The specified shape is drawn, within the number of dots specified for the range.



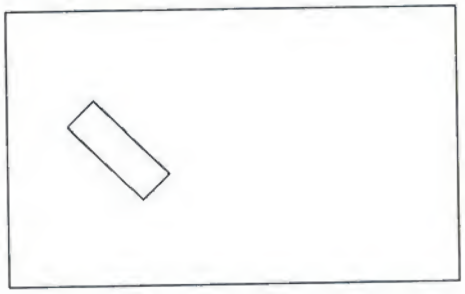
You can move an image in eight different directions.



It's also very easy to draw an zigzag line.

```
10 COLOR 15,1,1
20 SCREEN 2
30 DRAW "BM50,50"
40 DRAW "F60 G20 H60 E20"
50 GOTO 50
60 END
```

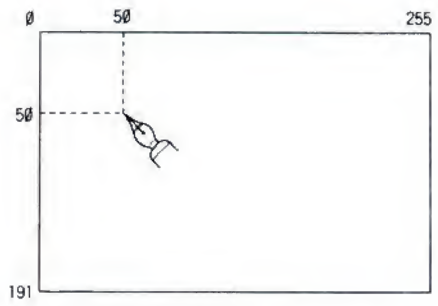
⇩ **RUN**



Line 10 defines the screen color.
Line 20 defines the screen mode.

```
30 DRAW"BM50,50"
```

This statement specifies the point where the drawing will start.



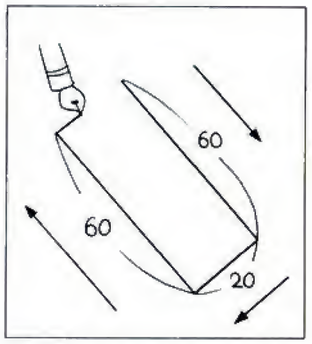
Even if you know the directions, if you don't know the starting point you can't draw anything.

Format

```
DRAW "BM X-coordinate, Y-coordinate"
```

```
4Ø DRAW "F6ØG2ØH6ØE2Ø"
```

A rectangle will be drawn starting at the point (5Ø,5Ø). The sides of the rectangle will be then be drawn 6Ø dots to the lower right. 2Ø dots to the lower left, 6Ø dots to the upper right and 2Ø dots to the upper right.



It's easy to change the length of a rectangle, but it's somewhat harder to change the angle.

Next lets draw a number of rectangles in a row.

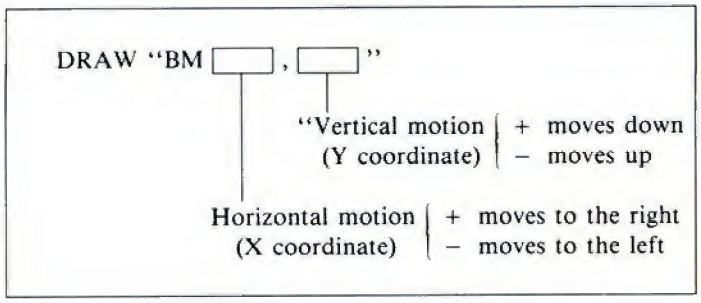
```
1Ø COLOR 15,1,1
2Ø SCREEN 2
3Ø DRAW"BM5Ø,5Ø"
4Ø FOR I=1 TO 2Ø
5Ø DRAW"BM+5,Ø"
6Ø DRAW"F6ØG2ØH6ØE2Ø"
7Ø NEXT I
8Ø GOTO 8Ø
9Ø END
```

The statements in line 4Ø FOR I = 1 TO 2Ø and line 7Ø NEXT I cause 2Ø rectangles to be drawn. However, they will all be drawn in the same position. In order to see each separate rectangle, each rectangle should be moved over a little bit from the position of the previous rectangle. This is done by the command in line 5Ø.

```
5Ø DRAW"BM+5,Ø"
```

This statement moves the starting point of each rectangle that is drawn, 5 dots to the right each time.

Format


DRAW "BM ☐ , ☐ "
"Vertical motion (Y coordinate) + moves down − moves up
Horizontal motion (X coordinate) + moves to the right − moves to the left

For example, the statement

```
50 DRAW"BM+5,+2"
```

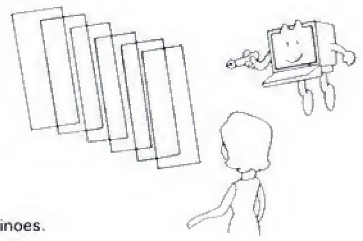
causes 20 rectangles which shift to the lower right side of the screen to be drawn.

In the DRAW command, instead of using parameters enclosed in quotes like F60G20H60E20, character variables can also be used.

```
DRAW "X[    ];"
        └character variable
```

```
 10 COLOR 15, 1,1
 20 SCREEN 2
 30 DRAW"BM50,50"
 40 FOR I=1 TO 20
 50 DRAW"BM+5,0"
 60 A$="F60G20H60E20"
 70 DRAW"XA$;"
 80 NEXT I
 90 GOTO 90
100 END
```

When this program is run it produces the same results as the program on page 70.

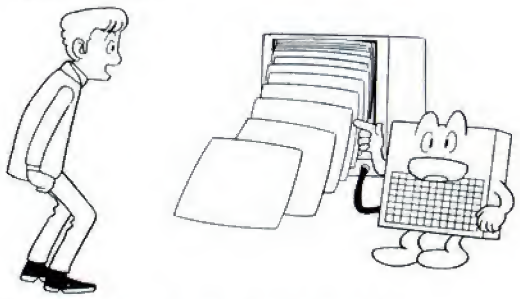

The rectangles are layered like a row of dominoes.

Moving the Sprite figure

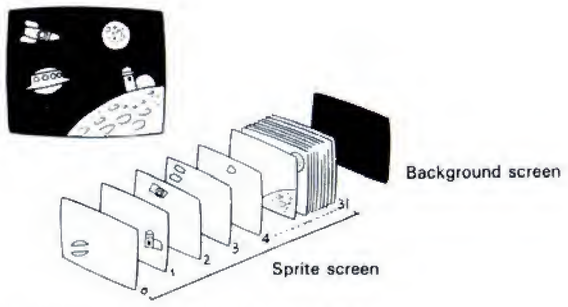
Up to now we have been studying how to draw shapes using the LINE and CIRCLE commands, although using these commands to move figures is somewhat difficult.
The SPRITE command is extremely useful for this purpose. Before studying the SPRITE command, lets learn a little bit more about the background, Sprite and boundary screens. In this case, the boundary screen is not used for screen display and will be omitted.

★ Background Screen
The background screen is used for graphics which do not move. Up to now, all of the LINE and CIRCLE commands have not been moved on the background screen. There are four modes for the background screen. The SCREEN command is used to change these modes.

★ Sprite Screen
The Sprite screen is used for the purpose of moving graphic figures. There are Ø to 31 Sprite screen numbers which can be attached to the Sprite screen. This means that there are 32 Sprite screens in all. The screen numbers are displayed in ascending order. Only one desired figure can be loaded at a time, from one screen to the next. However, if the Sprite screen number is changed, the same figure can be displayed on up to 32 screens.


This screen is only one layer, but there are actually 33 layers.

If nothing is written on the Sprite screen, it is transparent; therefore up to 33 layers of images can be viewed on the monitor.
Thus, when a figure drawn on a sprite screen at the back is over lapped by one drawn on a sprite screen nearer the front the overlapped portion of the figure at the back will disappear. This makes it possible to give display an appearance of depth.

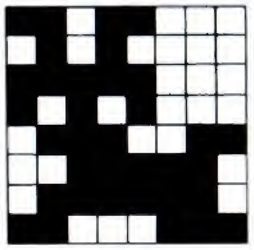


Sprite screens are transparent except where they contain figures.

In order to use the Sprite function, the following steps are necessary.

- Either 8 × 8 dot or 16 × 16 dot Sprite patterns must be defined.
- The data for characters to be displayed using the Sprite function must be defined in character strings.
- Both the Sprite screen number which the Sprite pattern will be displayed on, as well as the display size (standard or expanded) must be specified.
- Sprite patterns must be displayed on Sprite screens.

Let's make a Sprite pattern.
First an 8×8 dot figure must be defined.
In binary notation, a blank is expressed as Ø and a dark point as a 1.



	Binary		Hexadecimal		Decimal
→	&B11Ø11ØØØ	=	&HD8	=	216
→	&BØ1Ø1ØØØØ	=	&H5Ø	=	8Ø
→	&B11111ØØØ	=	&HF8	=	248
→	&B1Ø1Ø1ØØØ	=	&HA8	=	168
→	&BØ111ØØ11	=	&H73	=	115
→	&BØØ11111Ø	=	&H3E	=	62
→	&BØ111111Ø	=	&H7E	=	126
→	&B11ØØØ111	=	&HC7	=	199

A Sprite pattern can be written like this, by using a character string composed of eight character codes.
When the figure above is defined as a Sprite pattern, it is expressed by using a character string like this.

CHR$(216) + CHR$(8Ø) + CHR$(248) +
CHR$(168) + CHR$(115) + CHR$(62) +
CHR$(126) + CHR$(199)

If decimal notation is not used, either binary or hexadecimal expressions can be written as shown below.

Binary
CHR$(&B11Ø11ØØØ) + CHR$(&BØ1Ø1ØØØØ) + ...
.... CHR$(&B11ØØØ111)
Hexadecimal
CHR$(&HDH) + CHR$(&H5Ø) + ...
.... CHR$(&HC7)

When binary or hexadecimal expressions are used, the following symbols must be used before the numbers.

Binary	&B
Hexadecimal	&H

Up to 256 8×8 dot Sprite patterns can be defined on the MSX computer. Each of them is given a special Sprite pattern number.
Let's assign Sprite pattern number Ø to the pattern which was just created.

```
SPRITE$(Ø)=CHR$(216)+CHR$(8Ø) +
           CHR$(248)+CHR$(168)+
           CHR$(115)+CHR$(62)+
           CHR$(126)+CHR$(199)
```

SPRITE$

Command to define a sprite pattern

Format

```
SPRITE$ (pattern number)=figure expressed as a
                         character string
```

Sprite patterns can only be displayed in SCREEN modes 1 to 3. Sprite pattern sizes are determined as follows.

```
SCREEN,Ø........................... 8×8  dot standard
SCREEN,1........................... 8×8  dot expanded
SCREEN,2........................... 16×16 dot standard
SCREEN,3........................... 16×16 dot expanded
```

The PUT SPRITE command is used to display a Sprite pattern on the screen.

PUT SPRITE

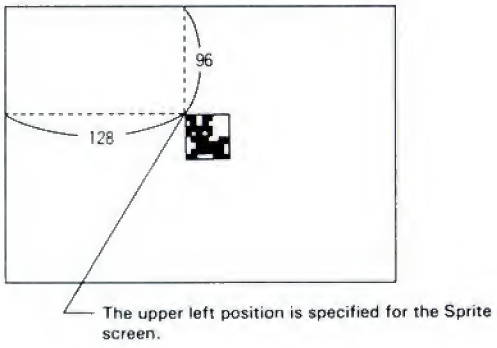
Command to display a sprite

Format

```
PUT SPRITE  Sprite screen number, Sprite position,
            color code, Sprite pattern number
```

```
 10 COLOR 15,4,7
 20 SCREEN 2,0
 30 FOR N=1 TO 8
 40 READ A : B$=B$+CHR$(A)
 50 NEXT N
 60 SPRITE$(0)=B$
 70 PUT SPRITE 0,(128,96),15,0
 80 GOTO 80
 90 END
100 DATA 216,80,240,168,115,62,126,199
```

⇩ **RUN**



The rabbit in the program above was displayed in white.

```
70 PUT SPRITE Ø,(128,96),15,Ø
```

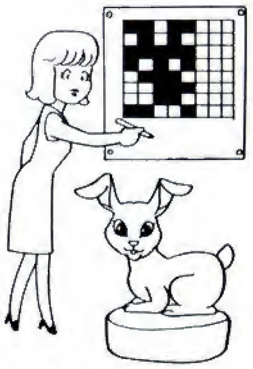
This command specified Sprite screen number Ø, position (128,96), color code 15, and pattern number Ø to be drawn.

If a Sprite pattern is not defined by the SPRITE$ command before executing the PUT SPRITE command, nothing will be displayed on the screen.

If line 2Ø is changed to

```
20 SCREEN 2,1
```

and the program is run the rabbit will be twice as big as before. In this case, the 8 × 8 Sprite dot pattern is expanded dot by dot to twice its original size.



The character string for the ear is 11Ø11ØØØ. Oh don't move! It just became 11ØØØØØØ.

Binary and Hexadecimal Conversions

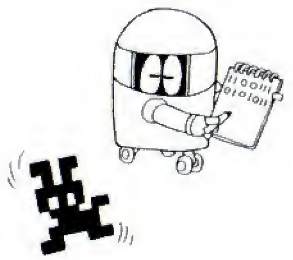
Binary Number	Decimal Number	Hexadecimal Number
Ø	Ø	Ø
1	1	1
1Ø	2	2
11	3	3
1ØØ	4	4
1Ø1	5	5
11Ø	6	6
111	7	7
1ØØØ	8	8
1ØØ1	9	9
1Ø1Ø	1Ø	A
1Ø11	11	B
11ØØ	12	C
11Ø1	13	D
111Ø	14	E
1111	15	F

- To convert a binary to a hexadecimal number

$$\underset{\underset{D}{\downarrow}}{\underline{11Ø1}}\,\underset{\underset{8}{\downarrow}}{\underline{1ØØØ}} \qquad \underset{\underset{5}{\downarrow}}{\underline{Ø1Ø1}}\,\underset{\underset{Ø}{\downarrow}}{\underline{ØØØ}}$$

- To convert a hexadecimal number to a decimal number

$$\begin{array}{ll} D & 8 = 13 \times 16 + 8 \times 1 = 216 \\ \downarrow & \downarrow \\ 16^1 & 16^0 \\ \| & \| \\ 16 & 1 \end{array}$$



The rabbit pattern is the same when it is represented by character strings written in binary, decimal or hexadecimal.

• To move a Sprite pattern

```
 10 COLOR 15,4,7
 20 SCREEN 2,1
 30 FOR N=1 TO 8
 40 READ A:B$=B$+CHR$(A)
 50 NEXT N
 60 SPRITE$(0)=B$
 70 PUT SPRITE 0,(0,90), 15,0
 72 C=INT(RND(1)*15)
 74 FOR I=1 TO 256
 76 PUT SPRITE 0,STEP(-1,0), C, 0
 78 NEXT I
 80 GOTO 72
 90 END
100 DATA 216,80,240,168,115,62,126,199
```

The parts which are underlined are rewritten or modified. When this program is executed, the rabbit will change colors and move from right to left. The program can be stopped and started again by pressing the STOP key.

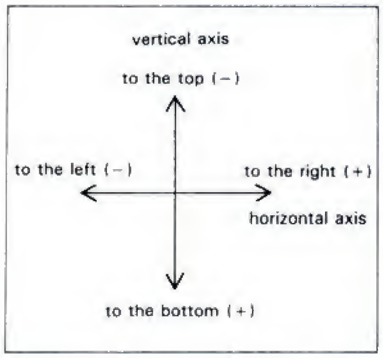
```
76 PUT SPRITE 0,STEP(-1,0), C,0
```

The rabbit can only be moved from position (0,90) by using the STEP command to indicate the variation range (−1,0).

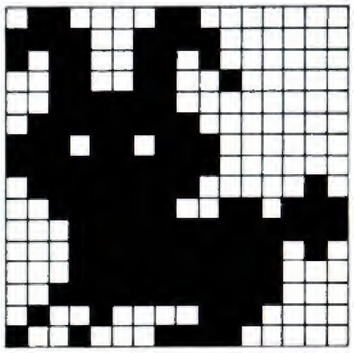
Format

```
STEP (horizontal variation, vertical variation)
```

Why don't you try changing the parameters in the STEP(−1,Ø) command of line 7Ø and observe the results?



- Creating 16×16 Dot Patterns
  The increase in the 16×16 pattern is quite evident. In binary notation blank points are expressed by Ø and dark points by 1.



I'm the binary rabbit. If you make me a decimal rabbit, I'll be light and easier to write.

Character strings in binary notation must be created in the order shown below.

Ø11ØØØØ1	1ØØØØØØØ
1111ØØ11	11ØØØØØØ
1Ø11ØØ11	Ø1ØØØØØØ
ØØ11ØØ11	ØØ1ØØØØØ
ØØ111111	.
Ø1111111	.
111Ø11Ø1	.
11111111	.
.	.
.	.
.	.
.	.
.	.
.	.
.	.
.	.

①	⑰
②	⑱
③	⑲
④	⑳
⑤	㉑
⑥	㉒
⑦	㉓
⑧	㉔
⑨	㉕
⑩	.
.	.
.	.
.	.
.	.
.	.
⑯	㉜

Character string DATA

```
 97, 243, 179,  51,  63, 127, 237, 255
127,  63,  31,  31,  31,  63,  72, 145
128, 192,  64,  32,   Ø, 128, 192, 192
13Ø,  55, 255, 25Ø, 248, 248, 112, 224
```

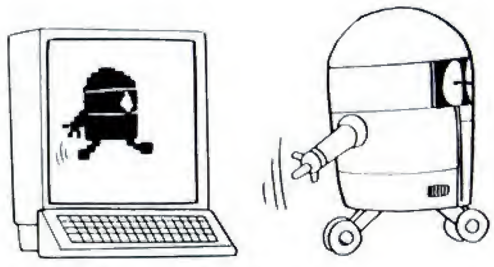
I'm those numbers.

Enter the following modifications to the previous program. Except for these changes, it should run the same as it did before.

```
 20 SCREEN 2, 2
 30 FOR N=1 TO 32

100 DATA · · · ·
```

```
 10 COLOR 15,4,7
 20 SCREEN 2,2
 30 FOR N=1 TO 32
 40 READ A : B$=B$+CHR$(A)
 50 NEXT N
 60 SPRITE$(0)=B$
 70 PUT SPRITE 0,(128,96), 15,0
 80 GOTO 80
 90 END
100 DATA 97, 243, 179, 51, 63, 127, 237, 255
110 DATA 127, 63, 31, 31, 31, 63, 72, 145
120 DATA 128, 192, 64, 32, 0, 128, 192, 192
130 DATA 130, 55, 255, 250, 248, 248, 112, 224
```


Next time let's make it like this!

ON SPRITE GOSUB

Format

```
ON SPRITE GOSUB line number
```

This command calls the subroutine starting at the specified line when the locating of figures on two sprite screens are the same.

SPRITE ON/OFF/STOP

Interruptions can be generated by using the SPRITE ON command.

Format

```
SPRITE ON
```

The subroutine specified by the ON SPRITE GOSUB command will then be executed.

```
SPRITE OFF
```

This command is used to prevent interrupts from being generated. After this command has been executed, the ON SPRITE GOSUB command can't be used until a SPRITE ON command is executed.

```
SPRITE STOP
```

This command stops Sprite interrupts. The difference between this and the SPRITE OFF command is that when t is stopped, it is evaluated whether or not there was an error in SPRITE. If there is an error when Sprite is stopped, and the SPRITE ON command is executed, an interrupt is immediately generated.

In this program, when Sprite screen number Ø and 1 are used and the program execution reaches line 14Ø, the buzzer sounds thirty times and the program stops.

```
 1Ø SCREEN 2,1
 2Ø A$ = CHR$(216) + CHR$(8Ø) + CHR$(24Ø) +
         CHR$(168) + CHR$(115) + CHR$(62) +
         CHR$(126) + CHR$(199)
 3Ø B$ = CHR$(56) + CHR$(124) + CHR$(56) +
         CHR$(254) + CHR$(56) + CHR$(56) +
         CHR$(68) + CHR$(13Ø)
 4Ø SPRITE$(Ø) = A$
 5Ø SPRITE$(1) = B$
 6Ø ON SPRITE GOSUB 13Ø
 7Ø SPRITE ON
 8Ø FOR I = Ø TO 191
 9Ø PUT SPRITE Ø,(Ø + I,96),1,Ø
1ØØ PUT SPRITE 1,(255 - I,96), 1Ø,1
11Ø NEXT I
12Ø END
13Ø SPRITE OFF
14Ø FOR J = 1 TO 3Ø: BEEP: NEXT J
15Ø RETURN 12Ø
```

Drawing character on the graphic screen

Up to now we've learned how to draw both text characters and graphics. However, the screen modes have been separated into text and graphic modes. Text and graphics couldn't be used together.
The next program shows how to write text characters on a graphic screen.

When you want to write text characters in the graphic mode, it must be specified first.

```
 10 SCREEN 2
 20 LINE(120,110)-(155,95),9
 30 LINE-(190,110),9
 40 LINE-(120,110), 9
 50 PAINT(160,105), 9
 60 LINE(135,140)-(175,110), 15,BF
 70 LINE(155,125)-(165,115), 5,BF
 80 LINE(140,140)-(150,120), 1,BF
 90 OPEN"GRP:" FOR OUTPUT AS #1
100 DRAW"BM120,70"
110 PRINT #1,"HOUSE"
120 CLOSE #1
130 GOTO 130
140 END
```

To write alphanumeric characters on a graphic screen execute the following command.

```
90 OPEN "GRP:" FOR OUTPUT AS #1
```

When this command is executed, the computer is informed that text characters can be written on the graphic screen.
After this, the PRINT command can be used just like it is in the text screen mode. The characters "HOUSE" are written by executing the command

```
110 PRINT #1,"HOUSE"
```

```
100 DRAW "BM 120,70"
```

This specifies the beginning of the display position.

STEP 5
MSX Music Class

Let's study some of the commands which make music. These commands are BEEP, PLAY and SOUND. However the SOUND command is somewhat difficult so it won't be covered here.

BEEP

This makes the buzzer sound.

Format

```
BEEP
```

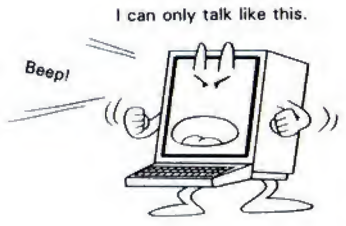
Beeps the speaker.

```
10 FOR I=1 TO 10
20 BEEP
30 NEXT I
40 END
```

⇩ **RUN**

```
The buzzer will ring 10 times
```

The BEEP command can't change the tone or length the tone will sound.



PLAY

This command performs music.

Format

```
PLAY "character string"
```

Music expressed by a character string is played when this command is executed.

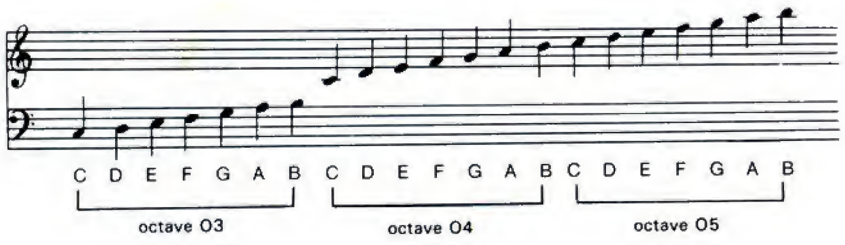
```
PLAY "CDEFRGAB" RETURN
```

This instruction plays "Do Re Mi Fa", pauses briefly and then plays "So La Ti".

The characters A to G each represent a note of the musical scale.

character	C	D	E	F	G	A	B
note	Do	Re	Mi	Fa	So	La	Ti

The character "R" represents a pause in the music. This is known as a rest. A melody can be played by entering PLAY, followed by the characters A to G or R enclosed in quotation marks.



• Specifying Octaves
The MSX computer can play eight octaves. In order to designate an octave use the letter "O" and a number (1 to 8). This must be written before the characters (A to G) which represent the tones of the scale.

```
PLAY "O3 CDEFGAB O4 CDEFGAB
        O5 CDEFGAB"  RETURN
```

This plays the scale for three octaves. Once a value for O is specified, the same value is maintained until a new value for O is encountered. When the computer is turned on the preset value for O is (O4).

• Writing Sharps and Flats
In order to be able to play a melody, it is also necessary to be able to play a half tone up (sharp) and a half tone down (flat).

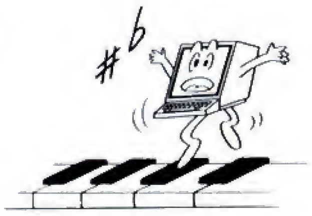
Sharps and Flats are written as follows.

A half tone up..................	# or + are written immediately after the characters A to G.
A half tone down..............	A minus sign "–" is written immediately after the characters A to G.

The symbols "#", "+", and "–" are the same as the black keys on the piano.

There are also some half tones which are exceptions.
C–is B, F–is E,B+ is C and E+ is F

Hey are there really that many black keys?



• Writing the length of a tone
If there is a tone length parameter after the characters representing the notes, those notes are played for the specified period of time. If there is no number following a note, that note is played for the length of time equal to the number following the character L

```
PLAY "C1D2E4F8R1G16A32B64" [RETURN]
```

The notes CDEF are gradually played faster and faster. After a rest the notes GAB are played even quicker. The tone length can be specified quite precisely by using a value from 1 to 64.

```
PLAY "C4D4E4F4G4A4B4"
```

It's rather inconvenient to specify the time value of a quarter note for each note. In cases like this, it's easier to write it out as shown below.

```
PLAY "L4 CDEFGAB"
```

Isn't it easier to write it out like that?
If there are numbers following the characters A to G and R, the tones will be played according to the indicated length. The letter "L" is used to specify the tone length. The value specified for L is maintained until a new value for L is encountered. When the computer is turned on the preset tone length value is "L4".

note name	whole note	1/2 note	1/4 note	1/8 note	1/16 note	1/32 note
musical symbol	𝅝	𝅗𝅥	♩	♪	𝅘𝅥𝅯	𝅘𝅥𝅰
value of L	1	2	4	8	16	32

• Dotted notes

A dotted note is a note followed by a dot. Its length is 1.5 times larger than the specified value for the tone length.
A period after the numeric value for the time length indicates a dotted note.

note name	dotted whole note	dotted 1/2 note	dotted 1/4 note	dotted 1/8 note	dotted 1/16 note	dotted 1/32 note
musical symbol	𝅝.	𝅗𝅥.	♩.	♪.	𝅘𝅥𝅯.	𝅘𝅥𝅰.
tone length	𝅝 + 𝅗𝅥	𝅗𝅥 + ♩	♩ + ♪	♪ + 𝅘𝅥𝅯	𝅘𝅥𝅯 + 𝅘𝅥𝅰	𝅘𝅥𝅰 + 𝅘𝅥𝅱
specified value	1.	2.	4.	8.	16.	32.

A dotted note is played just like it is written.

• Writing rests

A rest is a pause in the melody. Rests are written the same way notes are. The number following the R specifies the value of the rest.

rest name	whole rest	1/2 rest	1/4 rest	1/8 rest	1/16 rest	1/32 rest	1/64 rest
musical symbol	𝄻	𝄼	𝄽	𝄾	𝄿	𝅀	𝅁
value of R	1	2	4	8	16	32	64

• Setting the Tempo
The tempo within an entire piece will speed up and slow down at times. This is accomplished by using the T parameter followed by a number from (32 to 255). The larger numbers indicate faster tempos.
A value indicated for the parameter will be maintained until another T value is encountered. When the computer is turned on the preset value for the tempo is T120.

• Setting the volume
The V parameter and the numbers which follow (1 to 15), indicate the volume.

```
PLAY "L1V3CV6DV9EV12FV15G"  RETURN
```

The volume will gradually increase as the scale is played.
When V is specified there is no sound at all.
When the computer is turned on, the preset value for the volume is V8.

• Specifying the tone
To change the tone color, use either the S or M parameter followed by a number.
  S changes the envelope shape.
  M changes the envelope cycle.
The envelope can be specified by using the S and M parameters. The cycle pattern of the volume is changed when this occurs. At this time the vibrato characteristic is specified by the S parameter, and the vibrato speed is specified by the M parameter.
The S and M parameters should be used as a pair. The S and M parameters can not be used at the same time as the V (volume) parameter.

The following program changes the tone and plays the scale.

```
10 FOR S=0 TO 14
20 FOR M=1000 TO 10000 STEP 2000
30 PLAY "S=S;M=M;CDE"
40 NEXT M
50 NEXT S
60 END
```

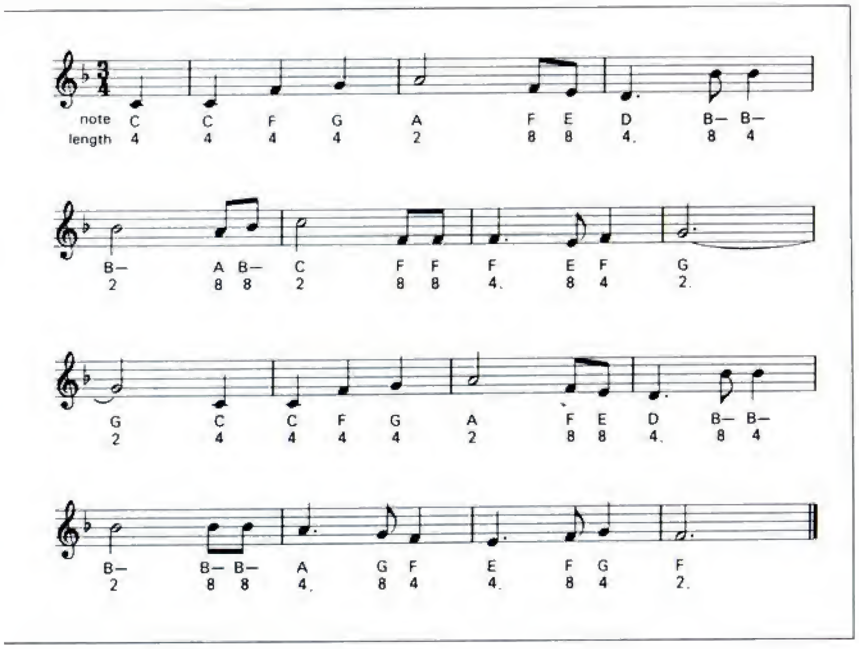
The value of the S parameter can be specified from 0 to 15.
The value of the M parameter can be specified from 1 to 65535.

```
PLAY "S10M3000 CDEFGAB"  RETURN
```

Try changing the numeric values which follow the S and M parameters to produce a variety of different tones.

Now let's make the MSX computer perform "Home on the Range".

```
10 PLAY "T128"
20 PLAY "O4L4CCFGA2L8FED4. B-B-4"
30 PLAY "B-2AB-O5C2O4FFF4. EF4G2."
40 PLAY "G2L4CCFGA2L8FED4. B-8B-4"
50 PLAY "B -2B-B-A4. GF4E4. FG4F2."
60 END
```



The MSX computer isn't all that simple. It can also play two and three note chords.
For example, to play C E G at the same time, place a comma between the notes of the chords as shown below.

```
PLAY "C","E","G"
```

Once you can play a scale, your repetoire will increase considerably.

Sample Programs

Sample 1

```
 10 REM COLOR GRAPHICS
 20 SCREEN 2
 30 COLOR 15, 1, 1
 40 C=1
 50 FOR X=35 TO 215 STEP 30
 60 FOR Y=35 TO 155 STEP 30
 70 PLAY "V14L64O6A"
 80 CIRCLE (X, Y), 30, C
 90 PAINT (X, Y), C
100 C=C+1
110 IF C>15 THEN C=1
120 NEXT Y
130 NEXT X
140 GOTO 140
150 END
```

When RUN is entered, this program draws circles in 15 colors across the entire screen. To stop the program, press CTRL + STOP .

Sample 2

```
10 REM KEYBOARD ORGAN
20 SCREEN 1
30 COLOR 1, 3, 3:CLS
40 LOCATE 4, 1:PRINT "KEYBOARD ORGAN"
50 PRINT "    ┌─┬─┬─┬─┬─┬─┬─┐"
60 PRINT "KEY │A│D│F│H│J│K│;│"
70 PRINT "    ├─┼─┼─┼─┼─┼─┼─┤"
80 PRINT "    │-│+│+│+│+│+│+│"
```

```
 90 PRINT "NOTE|B|C|D|F|G|A|C|"
100 PRINT "     └┴─┴─┴─┴─┴─┴─┘"
110 PRINT "     ┌┬─┬─┬─┬─┬─┬─┬─┬─┬─┐"
120 PRINT "KEY  |Z|X|C|V|B|N|M|,|.|/|"
130 PRINT "     ├┼─┼─┼─┼─┼─┼─┼─┼─┼─┤"
140 PRINT "NOTE|B|C|D|E|F|G|A|B|C|D|"
150 PRINT "     └┴─┴─┴─┴─┴─┴─┴─┴─┴─┘"
160 PRINT "PLAY NOTES BY PRESSING KEYS
    (Z-/, A-;)
170 A$=INKEY$:IF A$=" " THEN 170
180 IF A$=" Z " THEN PLAY "O3B"
190 IF A$=" X " THEN PLAY "O4C"
200 IF A$=" C " THEN PLAY "O4D"
210 IF A$=" V " THEN PLAY "O4E"
220 IF A$=" B " THEN PLAY "O4F"
230 IF A$=" N " THEN PLAY "O4G"
240 IF A$="M" THEN PLAY "O4A"
250 IF A$=" , " THEN PLAY "O4B"
260 IF A$=" . " THEN PLAY "O5C"
270 IF A$=" / " THEN PLAY "O5D"
280 IF A$=" A " THEN PLAY "O3B-"
290 IF A$=" D " THEN PLAY "O4C+"
300 IF A$=" F " THEN PLAY "O4D+"
310 IF A$=" H " THEN PLAY "O4F+"
320 IF A$=" J " THEN PLAY "O4G+"
330 IF A$=" K " THEN PLAY "O4A+"
340 IF A$=" ; " THEN PLAY "O5C+"
350 GOTO 170
360 END
```

To stop the program, press CTRL + STOP. (Press the keys in the uppercase mode.)

* The graphic symbols in lines 50 to 150 are only included to make the screen more readable, and may be omitted.

Sample 3

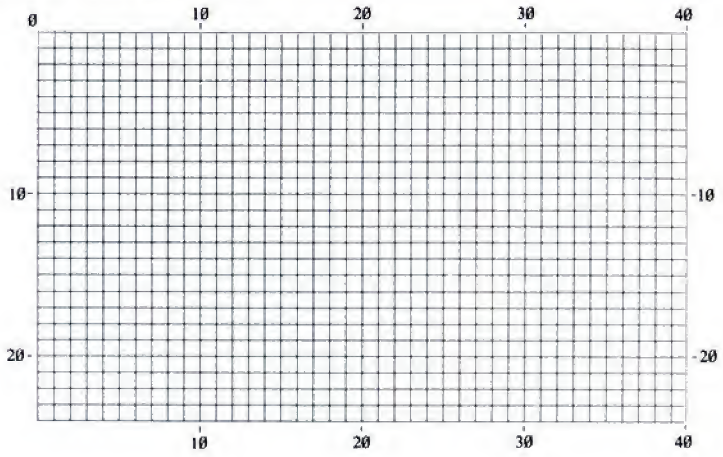
```
 10 REM ANIMATION
 20 SCREEN 2, 1
 30 COLOR 15, 1, 1:CLS
 40 LINE (  0, 114)-(255, 114), 2
 50 PAINT ( 50, 130), 2
 60 LINE ( 10,  50)-( 60, 114), 10, BF
 70 LINE ( 60, 114)-(110,  70), 9, BF
 80 LINE (110,  40)-(160, 114), 13, BF
 90 LINE (160, 114)-(200,  60), 12, BF
100 LINE (200,  30)-(255, 114), 14, BF
110 FOR I=1 TO 10
120 READ X, Y, X1, Y1
130 LINE (X, Y)-(X1, Y1), 1, BF
140 NEXT I
150 FOR I=1 TO 8
160 READ A:A$=A$+CHR$(A)
170 NEXT I
180 FOR I=1 TO 8
190 READ B:B$=B$+CHR$(B)
200 NEXT I
210 FOR I=1 TO 8
220 READ C:C$=C$+CHR$(C)
230 NEXT I
240 SPRITE$(0)=A$
250 SPRITE$(1)=B$
260 SPRITE$(2)=C$
270 X=0:Y=100
280 PLAY "S13M150F A F A F A F A F A F A"
290 PUT SPRITE 0, (X, Y), 8, 0
300 PUT SPRITE 1, (X, Y), 1, 1
310 PUT SPRITE 2, (X, Y), 15, 2
320 X=X+1
330 IF X>255 THEN X=0 ELSE 290
340 GOTO 280
350 END
```

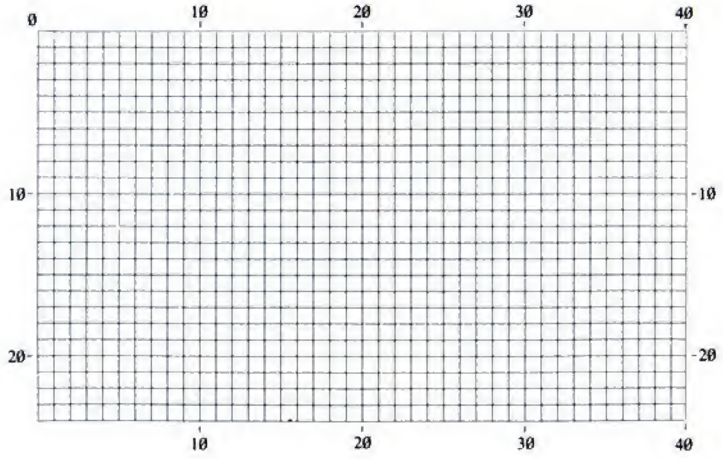
```
360 DATA  15,  60,  30,  80,  40,  60,  55,  80
370 DATA  65,  75,  75,  85,  80,  75,  90,  85
380 DATA  95,  75, 105,  85, 120,  50, 150,  60
390 DATA 120,  70, 150,  80, 165,  70, 175,  80
400 DATA 185,  70, 195,  80, 210,  40, 245,  50
410 DATA  16,   0,  32, 112,  32,   0,   0,   0
420 DATA   0,   0,   4,   0,   0,   0,  68,   0
430 DATA   0, 252, 216, 143, 233, 255,   0,   0
```

When RUN is input, an ambulance runs across the screen from left to right while sounding its siren. To stop the program, press CTRL + STOP .

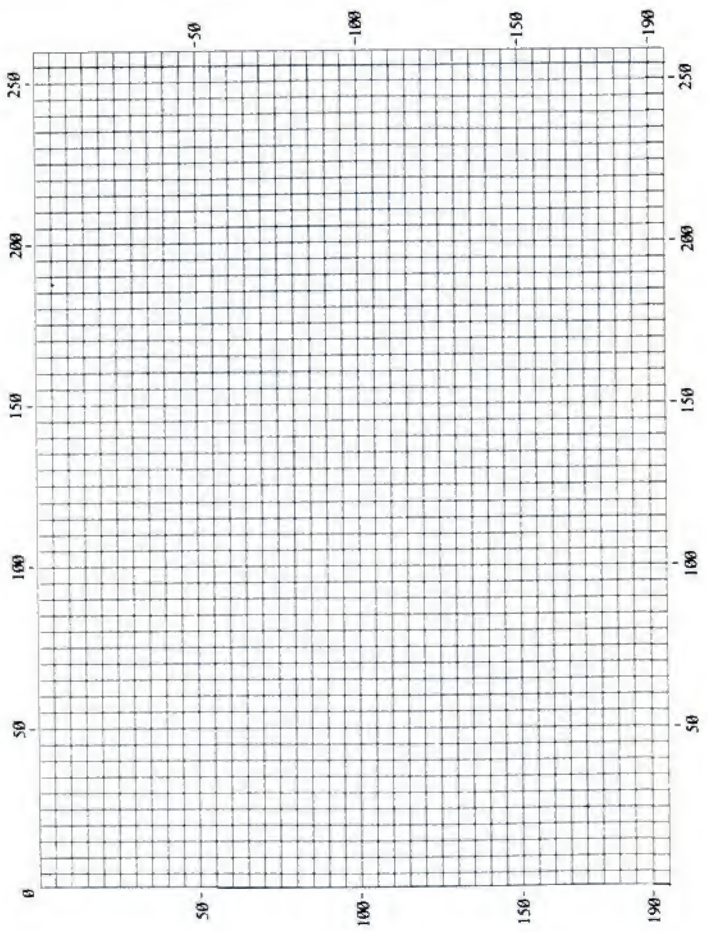
Design Sheets

Text Screen Design Sheets





Graphic Screen Design Sheets



Sprite Design Sheets (8 × 8)

Sprite Design Sheets (16 × 16)

INDEX

JVC
VICTOR COMPANY OF JAPAN, LIMITED

Printed in Japan
HC-BASIC-GB-A
Ⓐ1